TOMSK POLYTECHNIC UNIVERSITY ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский Томский политехнический университет» (ТПУ)

Школа – Инженерная школа ядерных технологий
Направление подготовки – 01.03.02 Прикладная математика и информатика
Отделение школы (НОЦ) – Отделение экспериментальной физики

БАКАЛАВРСКАЯ РАБОТА

Тема работы
Разработка интеллектуальной системы для прогнозирования даты проведения планово-предупредительных ремонтных работ

УДК 004.896:005.521:658.588.8

Студенты

Группа	ФИО	Подпись	Дата
0В8Б	Ильина Софья Андреевна		
0В8Б	Исмагилов Радик Рустемович		

Руководитель ВКР

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОЭФ	Семенов Михаил Евгеньевич	Кандидат ф. – м. наук, доцент		

Со-руководитель (по разделу «Концепция стартап-проекта»)

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ШИП	Попова Светлана Николаевна	Кандидат экономических наук, доцент		

КОНСУЛЬТАНТЫ:

По разделу «Социальная ответственность»

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ООД ШБИП	Сечин Андрей Александрович	Кандидат технических наук, доцент		

ДОПУСТИТЬ К ЗАЩИТЕ:

Руководитель ООП	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОЭФ ИЯТШ	Крицкий Олег Леонидович	Кандидат ф.– м. наук, доцент		

Томск – 2022 г.

TOMSK POLYTECHNIC UNIVERSITY ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский Томский политехнический университет» (ТПУ)

Школа – Инженерная школа ядерных технологий
Направление подготовки – 01.03.02 Прикладная математика и информатика
Отделение школы (НОЦ) – Отделение экспериментальной физики

УТВЕРЖДАЮ:

Руководитель ООП

_____ _____ Крицкий О.Л.

(Подпись) (Дата) (Ф.И.О.)

ЗАДАНИЕ
на выполнение выпускной квалификационной работы

В форме:

Бакалаврской работы

Студентам:

Группа	ФИО
0В8Б	Ильиной Софье Андреевне
0В8Б	Исмагилову Радику Рустемовичу

Тема работы:

Разработка интеллектуальной системы для прогнозирования даты проведения планово-предупредительных ремонтных работ	
Утверждена приказом директора (дата, номер)	

Срок сдачи студентом выполненной работы:	

ТЕХНИЧЕСКОЕ ЗАДАНИЕ:

Исходные данные к работе	Данные о структуре производственной площадки ООО «ЭПИ» (г. Москва), перечень объектов производства, данные, коррелирующие с реальными показателями телеметрии датчиков, установленных на конкретных элементах технологических объектов, таблица для обучения нейро-нечёткой системы на основе экспертных правил.

Перечень подлежащих исследованию, проектированию и разработке вопросов	1. Разработка базы данных характеристик технологических объектов производственных площадок 2. Разработка нейро-нечёткого конфигуратора для вычисления коэффициента неравномерности загрузки производственного оборудования 3. Разработка алгоритма прогнозирования остаточного ресурса деталей технологических объектов 4. Разработка алгоритма формирования оптимального графика планово-предупредительных ремонтных работ 5. Разработка алгоритма работы и архитектуры программного обеспечения для прогнозирования даты проведения планово-предупредительных ремонтных работ 6. Реализация программного продукта на языке программирования
Перечень графического материала	1. Схема структуры нейро-нечёткой сети 2. Графики, иллюстрирующие результаты работы интеллектуальной системы 3. Блок-схемы алгоритмов работы интеллектуальной системы 4. Диаграммы прецедентов в нотации UML 5. Диаграммы последовательности в нотации UML 6. ER-диаграмма инфологической модели базы данных в нотации Неймана 7. Рисунки, демонстрирующие графический интерфейс

Консультанты по разделам выпускной квалификационной работы

Раздел	Консультант
Концепция стартап-проекта	Попова Светлана Николаевна, доцент ШИП
Социальная ответственность	Сечин Андрей Александрович, доцент ООД ШБИП

Дата выдачи задания на выполнение выпускной квалификационной работы по линейному графику	

Задание выдал руководитель:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОЭФ ИЯТШ	Семенов Михаил Евгеньевич	к. ф.-м. н., доцент		

Задание приняли к исполнению студенты:

Группа	ФИО	Подпись	Дата
0В8Б	Ильина Софья Андреевна		
0В8Б	Исмагилов Радик Рустемович		

ЗАДАНИЕ ДЛЯ РАЗДЕЛА
«КОНЦЕПЦИЯ СТАРТАП-ПРОЕКТА»

Студенту:

Группа	ФИО
0В8Б	Ильиной Софье Андреевне
0В8Б	Исмагилову Радику Рустемовичу

Школа	ИЯТШ	**Направление**	01.03.02 Прикладная математика и информатика
Уровень образования	Бакалавриат		

Перечень вопросов, подлежащих разработке:	
Проблема конечного потребителя, которую решает продукт, который создается в результате выполнения НИОКР	*Сокращение эксплуатационных расходов предприятия*
Способы защиты интеллектуальной собственности	*Регистрация базы данных и исходного кода, патентование алгоритма и интерфейса*
Объем и емкость рынка	*Объём рынка СФО на год – 1947500 руб.*
Современное состояние и перспективы отрасли, к которой принадлежит представленный в ВКР продукт	*Прогнозируется рост объёма рынка*
Себестоимость продукта	*384 147 руб.*
Конкурентные преимущества создаваемого продукта	*Интеллектуальный алгоритм формирования графика ППР с минимизацией простоя и остаточной стоимости оборудования*
Сравнение технико-экономических характеристик продукта с отечественными и мировыми аналогами	*На основании конкурентных преимуществ*
Целевые сегменты потребителей создаваемого продукта	*Предприятия молочной и масложировой промышленности*
Бизнес-модель проекта	*Модель по А. Остервальду*
Производственный план	*6 ПО в первый год, далее увеличивающийся*
План продаж	*В первый год продаж: 2 996 353 рубля*
Перечень графического материала:	
При необходимости представить эскизные графические материалы (например, бизнес-модель)	*Модель по А. Остервальду, таблица сравнения характеристик*

Дата выдачи задания для раздела по линейному графику	

Задание выдал консультант по разделу «Концепция стартап-проекта» (со-руководитель ВКР):

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ШИП	Попова Светлана Николаевна	к. э. н., доцент		

Задание принял к исполнению студент:

Группа	ФИО	Подпись	Дата
0В8Б	Ильина Софья Андреевна		
0В8Б	Исмагилов Радик Рустемович		

ЗАДАНИЕ ДЛЯ РАЗДЕЛА
«СОЦИАЛЬНАЯ ОТВЕТСТВЕННОСТЬ»

Студентам:

Группа		ФИО	
0В8Б		Ильиной Софье Андреевне	
0В8Б		Исмагилову Радику Рустемовичу	
Школа	**Инженерная школа ядерных технологий**	**Отделение (НОЦ)**	**Экспериментальной физики**
Уровень образования	Бакалавриат	**Направление/ специальность**	01.03.02 Прикладная математика и информатика

Тема ВКР:

Разработка интеллектуальной системы для прогнозирования даты проведения планово-предупредительных ремонтных работ	
Исходные данные к разделу «Социальная ответственность»:	
Введение – Характеристика объекта исследования (вещество, материал, прибор, алгоритм, методика) и области его применения. – Описание рабочей зоны (рабочего места) при разработке проектного решения/при эксплуатации	*Объект исследования:* программное обеспечение для формирования графика проведения планово-предупредительных ремонтных работ оборудования на производственных площадках. *Область применения:* мониторинг и контроль промышленных автоматизированных систем. *Рабочая зона:* офисное помещение *Размеры помещения:*27,2 м^2 *Количество и наименование оборудования рабочей зоны:* 2 персональных компьютера *Рабочие процессы, связанные с объектом исследования, осуществляющиеся в рабочей зоне:* алгоритмическая и программная разработка с использованием персонального компьютера
Перечень вопросов, подлежащих исследованию, проектированию и разработке:	
1. Правовые и организационные вопросы обеспечения безопасности при разработке проектного решения: – специальные (характерные при эксплуатации объекта исследования, проектируемой рабочей зоны) правовые нормы трудового законодательства; – организационные мероприятия при компоновке рабочей зоны.	– Рабочее место при выполнении работ сидя регулируется ГОСТом 12.2.032-78 – Организация рабочих мест с электронно-вычислительными машинами регулируется СанПиНом 2.2.2/2.4.1340-03 – Трудовой кодекс Российской Федерации: федер. Закон от 30 дек. 2001 г. №197-ФЗ Раздел 10
2. Производственная безопасность при разработке проектного решения: – Анализ выявленных вредных и опасных производственных факторов	– Отклонение показателей микроклимата; – Недостаточная освещённость рабочей зоны; – Пониженная световая и цветовая контрастность; – Повышенный уровень шума на рабочем месте; – Повышенный уровень статического электричества; – Повышенная запыленность воздуха рабочей зоны; – Опасность поражения электрическим током;
3. Экологическая безопасность при разработке проектного решения:	Анализ воздействия на литосферу: – Утилизация компьютеров, оргтехники и бумаги; Анализ воздействия на гидросферу: – Производство компьютерной техники; Анализ воздействия на атмосферу: – Выделение вредных веществ при нагреве материнской платы; – Повышенная сухость воздуха при работе компьютера;

4. Безопасность в чрезвычайных ситуациях при разработке проектного решения:	– Затопление; – Землетрясение; – Короткое замыкание проводки; – Наиболее типичная ЧС: Пожар;

Дата выдачи задания для раздела по линейному графику	

Задание выдал консультант:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ООД ШБИП	Сечин Андрей Александрович	к.т.н.		

Задание принял к исполнению студент:

Группа	ФИО	Подпись	Дата
0В8Б	Ильина Софья Андреевна		
0В8Б	Исмагилов Радик Рустемович		

Реферат

Выпускная квалификационная работа содержит 151 страницу, 29 рисунков, 16 таблиц, 31 источник, 5 приложений.

Ключевые слова: нейро-нечёткие сети, адаптивная нейро-нечёткая система вывода, прогнозирование на основе анализа данных, комбинаторные задачи оптимизации, информационные системы.

Цель работы – создание интеллектуальной системы для прогнозирования даты проведения планово-предупредительных ремонтных работ на производственном оборудовании.

В результате работы была разработана интеллектуальная система для прогнозирования даты проведения планово-предупредительных ремонтных работ на производственном оборудовании.

Для обработки были взяты данные о структуре производственной площадки ООО «ЭПИ» (г. Москва), перечень объектов производства, данные, коррелирующие с реальными показателями телеметрии датчиков, установленных на конкретных элементах технологических объектов, таблица для обучения нейро-нечёткой системы на основе экспертных правил.

Для разработки программного модуля был использован язык программирования Python, для хранения данных была выбрана СУБД PostgreSQL. Графический интерфейс программного обеспечения был реализован с помощью фреймворка Qt. Модель машинного обучения была написана на основе фреймворка PyTorch.

Оглавление

Определения, обозначения, сокращения, нормативные ссылки

Авторизация – предоставление определённому лицу или группе лиц прав на выполнение определённых действий.

База данных (БД) – совокупность специальным образом организованных данных, хранимых в памяти вычислительной системы, отображающих состояние объектов и их взаимосвязей в рассматриваемой предметной области.

БИД/ЗИП – быстро изнашиваемые детали, запчасти и приборы.

ННС – нейро-нечёткая сеть.

ППР – планово-предупредительный ремонт.

СУБД – система управления базами данных.

Фреймворк – программное обеспечение, позволяющее автоматизировать разработку и тестирование программного продукта.

ANFIS – Adaptive Network Fuzzy Inference System – адаптивная сетевая система нечёткого вывода.

GUI (Graphic user interface) – графический интерфейс пользователя.

Matplotlib – библиотека на языке программирования Python для визуализации данных двумерной и трёхмерной графикой.

NumPy – библиотека для работы с многомерными массивами и высокоуровневыми математическими функциями на языке Python.

Python – высокоуровневый язык программирования общего назначения

PyTorch – фреймворк машинного обучения для языка Python.

PostgreSQL – свободная объектно-реляционная система управления базами данных.

Qt – фреймворк для разработки кроссплатформенного программного обеспечения.

Введение

Эксплуатационные расходы промышленных предприятий в зависимости от масштабов производства могут составлять от 15 до 30% от общего числа затрат [1]. Объём закупок запасных изделий имеет непосредственное влияние на данный показатель и напрямую зависит от оценки состояния оборудования, а также плана ремонтов компании.

В настоящий момент на многих промышленных предприятиях используется экспертный метод оценки состояния оборудования, либо на основании эксплуатационного срока службы, предоставленного производителем агрегатов. Однако данный не позволяет учитывать реальную нагрузку аппаратов, что приводит либо к авариям на производстве, либо к преждевременным заменам элементов оборудования.

Система планово-предупредительного ремонта (ППР) является наиболее надёжным подходом к организации и планированию технического обслуживания и ремонтов оборудования на производственных площадках [2]. Данный метод позволяет поддерживать заданный уровень исправности и работоспособности оборудования, т.к. предполагает проводить замены деталей оборудования без учёта его реального состояния. Однако существенным недостатком системы ППР является высокая стоимость обслуживания и рост эксплуатационных расходов, что делает метод экономически неэффективным при отсутствии должного обоснования.

Целью работы является разработка интеллектуальной системы для прогнозирования даты проведения планово-предупредительных ремонтных работ на производственном оборудовании. Для её достижения необходимо выполнить ряд задач:

- Разработать нейро-нечёткую сеть для прогнозирования коэффициента неравномерности загрузки производственного оборудования;

- Реализовать алгоритм вычисления остаточного ресурса и необходимого запаса сменных деталей на основании коэффициента неравномерности загрузки;
- Разработать алгоритм формирования оптимального графика ППР с минимизацией суммарной остаточной стоимости оборудования и стоимости простоя на производстве;
- Реализовать программный продукт с графическим интерфейсом пользователя.

Работа выполнена в рамках договора возмездного оказания услуг с индустриальным партнёром АО «УК ЭФКО». Авторы выражают благодарность Алексею Леонидовичу Руцкову – начальнику отдела оптимизации бизнес-процессов филилала АО «УК ЭФКО» в г. Воронеж, Екатерине Анатольевне Мироненко – специалисту оптимизации бизнес процессов отдела оптимизации бизнес-процессов филиала АО «УК ЭФКО» в г. Воронеж, Андрею Игоревичу Бондаренко – специалисту по созданию информационных систем отдела оптимизации бизнес-процессов филиала АО «УК ЭФКО» в г.Воронеж и Юрию Николаевичу Барзанову – специалисту по оптимизации цепей поставок отдела оптимизации бизнес-процессов филиала АО «УК ЭФКО» в г. Воронеж за участие в постановке задачи, сотрудничестве при разработке и обсуждении результатов работы.

Результаты работы представлены на таких конференциях как:

- Международная научно-практическая конференция «Электронные средства и системы управления» (Томск, 18.11.2021),
- XIX Международная конференция студентов, аспирантов и молодых учёных «Перспективы развития фундаментальных наук» (Томск, 28.04.2022).

И опубликованы в:

- Материалах докладов XVII Международной научно-практической конференции «Электронные средства и системы управления»,

- Journal of Physics: Conference Series 2022 (в печати),
- Сборнике научных трудов XIX Международной конференции студентов, аспирантов и молодых ученых "Перспективы развития фундаментальных наук" (в печати).

1. Модуль прогнозирования коэффициента загрузки

1.1. Описание проблемы

Необходимость оценки и прогнозирования состояния производственного оборудования возникает из-за того, что его эксплуатация происходит под влиянием многочисленных факторов. Эти факторы обусловлены конструктивными и функциональными особенностями оборудования, а также случайным внешним воздействием. Всё это влияет на постепенное ухудшение технического состояния оборудования во времени, что может привести к неисправности оборудования. В связи с этим полезно заранее прогнозировать или оценивать выход из строя оборудования и, таким образом, проводить планово-предупредительный ремонт. Для диагностики и прогнозирования потенциальной неисправности оборудования применяются различные методы и модели проактивной поддержки принятия решений [3].

Экспертный подход, как способ учёта загрузки оборудования на производстве, часто не учитывает фактические режимы работы и, как следствие, реальную загрузку. В литературе [3, 4] предложено множество подходов для диагностики технического состояния и прогнозирования остаточного ресурса промышленного оборудования, которые могут позволить уточнить фактическую загрузку. Подходы можно разделить на три больших класса: а) основанных на физических моделях (physical-based methods, PhM), б) использовании формализованных знаний (knowledge-based methods, KBM), в) анализе данных (data-driven methods, DDM).

Первый класс подходов используют известные математические модели для описания процессов износа оборудования. Построение адекватной физической модели даёт хорошие результаты оценки и прогнозирования, но сопряжено с множеством сложностей: для точной модели требуется глубокое понимание того, как работает оборудование и самих механизмов его износа, к тому же метод основан на ручной процедуре построения математической модели, что очень трудоёмко и занимает много времени. Подход, основанный на использовании формализованных знаний, использует опыт и знания экспертов для решения

задач поддержки принятия решений. Метод подразумевает наличие определённого набора правил или функций отношений, определение которого может быть трудоёмким процессом. Подход, основанный на анализе данных, моделирует состояние оборудования на основе собранных в процессе мониторинга данных. Этот подход обладает наибольшим потенциалом и эффективностью, так как обладает свойствами универсальности и не требует глубоких априорных знаний [4].

1.2. Постановка задачи

Учитывая обозначенную проблему и известные подходы прогнозирования остаточного ресурса производственного оборудования, в данной работе мы предлагаем решение, основанное на анализе данных – интеллектуальная система на базе нейро-нечёткой сети. Программная реализация разрабатываемого решения включает адаптивную нейро-нечёткую систему вывода (Adaptive Neuro-Fuzzy Inference System, ANFIS), которая сочетает математический аппарат нечёткой логики и моделей нейронных сетей. Предложенная гибридная интеллектуальная модель сочетает преимуществами обоих методов.

1.3. Требования к разрабатываемой системе

Разрабатываемое решение должно прогнозировать коэффициент загрузки оборудования, исходя из которого вычисляется остаточный ресурс, по которому можно составить оптимальный график ремонтных работ. В связи с этим, основное требование, предъявляемое к системе, это точность. Так, в области диагностики технического состояния оборудования, приемлемой считается процентная ошибка прогноза не более 10%. В нашем случае индустриальный партер выдвинул требование: 3-5%.

Также решение должно быть достаточно гибким, для чего необходимо реализовать возможность задания пользователем следующих параметров нейро-нечёткой сети (ННС):

– произвольной структуры ННС, в том числе числе возможность настройки пользователем количества входов, количества скрытых слоёв, количества нейронов на них, произвольное задание функций активации;
– алгоритмов обучения, в том числе Мамдани, Сугено или Такаги-Сугено-Канга [5];
– произвольного числа лингвистических правил обучения в соответствии с алгоритмами Мамдани и Сугено;
– графической формы отображения настроек ННС, результатов её работы;
– типа и количества функций принадлежности;
– количества эпох обучения.

Нейро-нечёткая сеть – выполняющая функцию блока прогнозирования, должна также в автоматическом режиме взаимодействовать с блоком определения остаточного ресурса и с базой данных.

1.4. Структура нейро-нечёткой сети

Разработанная ННС имеет нечёткую и линейную части. Нечёткая часть представлена ANFIS моделью, имеющей структуру многослойной нейронной сети из пяти основных слоёв, каждый из которых выполняет свою функцию (рис. 1.1, слева).

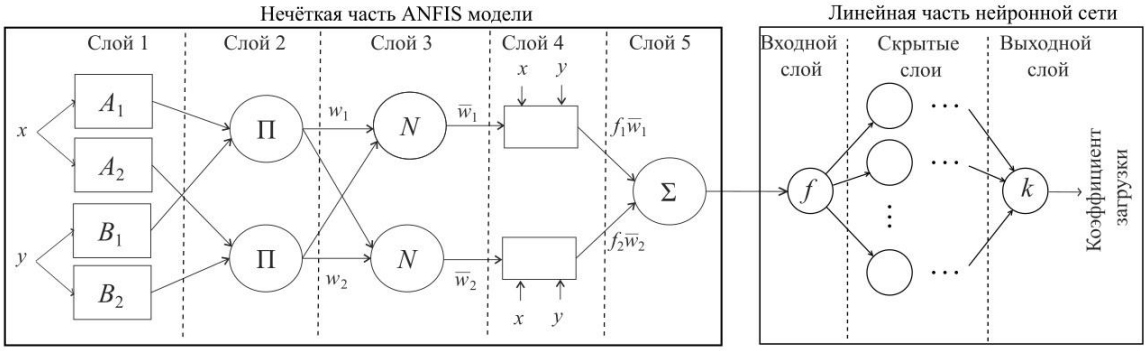


Рисунок 1.1 – Общая схема нейро-нечёткой сети на основе ANFIS модели

Слой 1 получает входные параметры x, y каждый из которых представлен отдельным нейроном и вводит их в модель ANFIS. Этот слой считается входом нечёткой системы, каждый его узел является адаптивным узлом с функцией принадлежности $\mu_{A_i}(x), \mu_{B_i}(y), i = 1,2$. Выход *Слоя 1* становится входом *Слоя 2*,

несущего предыдущие значения функций принадлежности П, которые распределяются на основе входных значений. Узлы *Слоя 2* определяют нечёткие правила и передают их на *Слой 3* сети с соответствующей степенью активности $\omega_i = \mu_{A_i}(x) \cdot \mu_{B_i}(y), i = 1,2$. Затем, в узлах *N Слоя 3* нормализуются степени активности правил $\overline{\omega_i} = \frac{\omega_i}{\omega_1 + \omega_2}, i = 1,2$. *Слой 4* (слой консеквентов) принимает узлы *N*, функции П и входы:

$$f_i \cdot \overline{\omega_i} = (p_i x + q_i y + r_i) \cdot \overline{\omega_i}, i = 1,2,$$

где p_i, q_i, r_i – параметры консеквента. После чего представляется модель первого порядка на основе производных параметров, и отправляется на выходной *Слой 5* (сумматор Σ) [6]:

$$f = \sum_i f_i \overline{\omega_i}.$$

Линейная часть (рис. 1.1, справа) содержит *Входной слой* с одним нейроном, несколько *Скрытых слоёв* с произвольным количеством нейронов на каждом из них, и *Выходной слой* с одним нейроном. *Входной слой* принимает выход с сумматора Σ на *Слое 5* и вводит данные на первый *Скрытый слой*. Каждый скрытый слой применяет различные преобразования к входным данным и передаёт результаты на следующий. *Выходной слой* принимает выход с последнего *Скрытого слоя* и преобразовывает данные в одномерный вектор.

Для некоторых параметров описанной сети реализована возможность пользовательской настройки. Так, в нечёткой части ННС пользователь может настроить количество нейронов на *Слое 1*, в соответствии с имеющимися данными. Далее, настраиваются типы функций принадлежности для каждого входа и их количество, после чего можно выбрать алгоритм формирования следствий. В стандартной реализации ANFIS используется алгоритм Такаги-Сугено-Канга, но можно выбрать алгоритм Мамдани или Сугено. В линейной части пользователь может выбрать количество скрытых слоёв и нейронов на них, а также типы функций активации.

Для обучения нечёткой и линейно частей разработанной ННС в данной работе используется алгоритм обратного распространения ошибки. В алгоритме обратного распространения различают два прохода, выполняемых в процессе вычислений: первый проход называют прямым (forward pass), второй – обратным (back pass). При прямом проходе (forward pass) синаптические веса остаются неизменными во всей сети, но последовательно вычисляются функциональные сигналы. Реализован алгоритм обратного распространения с использованием среднеквадратичной ошибки для настройки параметров. В качестве оптимизатора используется устойчивый алгоритм обратного распространения.

1.5. Формирование правил

Для обучения ANFIS модели требуется использование алгоритмов нечёткого логического вывода, которые базируются на понятии нечёткого множества, т. е. некоторого множества A в некотором непустом пространстве X, которое представимо множеством пар:

$$A = \left\{ \left\langle x, \mu_A(x) \right\rangle \mid x \in U \right\}, \tag{1.1}$$

где $\mu_A : U \to [0,1]$ – функция принадлежности нечёткого множества A, показывающая степень принадлежности каждого элемента x множеству A [7].

Понятие нечёткого логического вывода подразумевает процесс, в результате которого из нечётких предпосылок получаются некоторые следствия. В общем случае, прямой нечёткий логический вывод, применяющийся в данной реализации адаптивной нейро-нечёткой системы, имеет следующие этапы:

1. Задание нечёткой импликации вида: ЕСЛИ x есть A, ТО y есть B, где x – входная переменная, $x \in X$, X – область определения предпосылки нечёткого продукционного правила; A – нечёткое множество, определённое на X, с функцией принадлежности $\mu_A(x) \to [0,1]$, y – выходная переменная, $y \in Y$, где Y – область определения подзаключения нечёткого продукционного правила, B – нечёткое

множество, определённое на Y, с функцией принадлежности $\mu_B(y) \to [0,1]$.

2. Задание нечётких подусловий вида «x` есть $A$`», где x` – фактическое значение переменной $x$, $A$` – нечёткое множество, отражающее значение x`.
3. Задание нечётких подзаключений вида «y` есть $B$`», где y` – фактическое значение переменной $y$, $B$` – нечёткое множество, отражающее значение y` [7].

Для нечёткого логического вывода предложены различные алгоритмы [5]. В представленной реализации использована система нечёткого вывода Такаги-Сугено-Канга. Общая форма модели может быть представлена в векторном виде:

$$\text{ЕСЛИ } x \text{ это } A, \text{ТО } y = f(x),$$

где $f(x) = f(x_1, x_2, ..., x_N)$ – вещественная функция нескольких переменных.

В качестве функции заключения обычно используют линейный полином:

$$y_i = p_{i0} + \sum_{j=1}^{N} p_{ij} x_j, \tag{1.2}$$

где p_{ij} – коэффициенты, настраиваемые в процессе обучения сети, условия, представляются функциями принадлежности [8, 9].

Для работы с пользовательскими правилами реализованы алгоритмы Мамдани и Сугено. На вход этих алгоритмов принимаются некоторые входные данные (переменные x и y), и к ним применяются заданные правила вида:

если x есть A, и y есть B, то z есть C – для алгоритма Мамдани,

если x есть A, и y есть B, то $z = ax + by$ – для алгоритма Сугено,

на выходе получают значение прогнозируемой переменной z.

Опишем работу обоих алгоритмов.

Алгоритм Мамдани. Предположим, что базу знаний образуют два нечётких правила:

$П_1$: если x есть A_1 и y есть B_1, то z есть C_1,

Π_2: если x есть A_2 и y есть B_2, то z есть C_2,

где x и y – имена входных переменных, z – имя переменной вывода, A_1, A_2, B_1, B_2, C_1, C_2 – некоторые заданные функции принадлежности. Алгоритм Мамдани математически может быть описан следующим образом:

1. Нечёткость: находятся степени истинности для предпосылок каждого правила $A_1(x_0)$, $A_2(x_0)$, $B_1(y_0)$, $B_2(y_0)$.
2. Нечёткий вывод: находятся уровни отсечения для предпосылок каждого из правил

$$\alpha_1 = A_1(x_0) \wedge B_1(y_0),$$

$$\alpha_2 = A_2(x_0) \wedge B_2(y_0).$$

Затем находятся усечённые функции принадлежности:

$$C'_1(z) = (\alpha_1 \wedge C_1(z)),$$

$$C'_2(z) = (\alpha_2 \wedge C_2(z)).$$

3. Композиция: производится объединения найденных усечённых функций, что приводит к получению итогового нечёткого подмножества для переменной выхода с функцией принадлежности

$$\mu_\Sigma(z) = C(z) = C'_1(z) \vee C'_2(z) = (\alpha_1 \wedge C_1(z)) \vee (\alpha_2 \wedge C_2(z)).$$

4. Приведение к чёткости (нахождение z_0): может производиться разными методами, например центроидным:

$$z_0 = \frac{\int_\Omega z\mu_\Sigma(z)dz}{\int_\Omega \mu_\Sigma(z)dz}.$$

Алгоритм Сугено. Пусть есть база знаний:

Π_1: если x есть A_1 и y есть B_1, то $z_1 = a_1x + b_1y$,

Π_2: если x есть A_2 и y есть B_2, то $z_2 = a_2x + b_2y$.

1. Первый этап – как в алгоритме Мамдани.

2. Находятся уровни отсечения для предпосылок каждого правила как в алгоритме Мамдани и индивидуальные выходы правил по формулам:

$$z_1^* = a_1 x_0 + b_1 y_0,$$

$$z_2^* = a_2 x_0 + b_2 y_0.$$

3. Определяется чёткое значение переменной вывода [10]:

$$z_0 = \frac{\alpha_1 z_1^* + \alpha_2 z_2^*}{\alpha_1 + \alpha_2}.$$

1.6. Реализация функций принадлежности

На *Слое 2* сети ANFIS формируются правила на основе всех возможных комбинаций функций принадлежности, которые необходимы для задания нечёткой импликации. Функции принадлежности могут быть разного вида, например, кусочно-линейного, как трапециевидная функция:

$$f_T(x,a,b,c,d) = \begin{cases} 0, x \le a \\ \dfrac{x-a}{b-a}, a \le x < b \\ 1, b \le x < c \\ \dfrac{d-x}{d-c}, c \le x < d \\ 0, d \le x \end{cases}, \quad (1.3)$$

где $a \le b \le c \le d \in \mathbb{R}$ [11]. Примером функций принадлежности S-образного и Z-образного (в зависимости от положительного или отрицательного значения параметра a соответственно) видов может служить сигмоидальная функция, задающаяся следующим аналитическим выражением:

$$f_S(x,a,b) = \frac{1}{1+e^{-a(x-b)}}, \quad (1.4)$$

где $a < b \in \mathbb{R}$ [11]. В качестве функций П-образного вида часто применяется функция Гаусса:

$$f_G(x,\mu,\sigma) = e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (1.5)$$

где μ – математическое ожидание, а σ^2 – дисперсия распределения [11].

Все вышеописанные функции принадлежности реализованы в данной работе. Для пользовательской настройки нечёткой части ННС для каждого входного нейрона можно выбрать тип функции принадлежности, а также их количество. Далее, все входные числовые параметры этих функций принадлежности настраиваются автоматически исходя из вида входных данных конкретного нейрона.

1.7. Алгоритм работы интеллектуальной системы

Вышеописанная нейро-нечёткая сеть прогнозирует вектор коэффициентов загрузки оборудования во времени по входным данным с датчиков системы SCADA (Supervisory Control And Data Acquisition – диспетчерское управление и сбор данных), оперативных отчётов и имитационной модели. Коэффициент неравномерности загрузки показывает, насколько больше или меньше номинального уровня загрузки использовалось оборудование на самом деле. Но для планово-предупредительного ремонта более важным параметром является необходимый запас быстроизнашиваемых деталей, запчастей и приборов (БИД/ЗИП), который, в предложенном решении, определяется в зависимости от количества установленных позиций однотипных БИД/ЗИП и от их среднего остаточного ресурса.

Значение среднего остаточного ресурса можно вычислить по формуле:

$$R = \left(1 - k_{загрузки} \cdot \frac{t_2}{t_1}\right) \cdot 100\% \qquad (1.6)$$

где t_1 – текущее время эксплуатации рассматриваемого оборудования, t_2 – нормативное время эксплуатации для рассматриваемого оборудования, т.е. такое время, за которое оборудование износится в предположении, что в течении этого времени оно имеет равномерную нормативную загрузку, $k_{загрузки}$ – коэффициент загрузки оборудования, в данной реализации определяемый ННС. По остаточному ресурсу определяется оптимальный запас заменяемых деталей оборудования, который нужно иметь на складе, а также оптимальную дату проведения планово-предупредительных ремонтных работ. Заметим, что

формула (1.6) аналитическая, что удобна тем, что при её использовании применяются только линейные операции над векторами.

После вычисления значений остаточного ресурса оборудования следует найти необходимый запас БИД/ЗИП входящих в состав этого оборудования. Сначала вводится предположение, что все БИД/ЗИП входящие в состав некоторого оборудования имеют такие же значения остаточного ресурса, как и оборудование в целом, т.е. износ происходит равномерно по всем элементам оборудования. Исходя из этого предположения вектор остаточного ресурса оборудования распространяется на входящие в него БИД/ЗИП.

Необходимый запас БИД/ЗИП нужно вычислять, учитывая то, что на производстве могут использоваться однотипные БИД/ЗИП в составе разного оборудования. Поэтому значения остаточного ресурса усредняются по однотипным БИД/ЗИП, тем самым получаем $\overline{R}$.

Нормативное запас БИД/ЗИП устанавливается исходя из количества позиций однотипного оборудования в соответствии с таблицей 1.1.

Таблица 1.1 – Зависимость нормативного запаса БИД/ЗИП от количества позиций однотипного оборудования

Количество позиций однотипного оборудования, шт	**Нормативный запас БИД/ЗИП, % от установленного количества комплектующих**
1	100
2	50 – 100
3 – 10	30 – 50
11 – 100	10 – 20
101 и более	5 – 10

Изначально принимаем, что зависимость в таблице верна при остаточном ресурсе оборудования $\overline{R} = 0.5$. Для того чтобы уточнить запас БИД/ЗИП исходя из действительных значений остаточного ресурса, применяем формулу:

$$\left(\overline{R} - R_{норм.}\right) \cdot a_N \cdot N \qquad (1.7)$$

где $R_{норм.}$ – такой остаточный ресурс оборудования, при котором значения нормативного запаса БИД/ЗИП (табл. 1.1) считаются достоверными, изначально

$R_{норм.} = 0.5$, но в процессе применения формулы (1.7) к вектору остаточного ресурса он может варьироваться в диапазоне (0; 1), если значения разности $\left(\overline{R} - R_{норм.}\right)$ становятся недопустимыми (например меньше 0), a_N – диапазон нормативного запаса в зависимости от количества установленных позиций однотипных БИД/ЗИП (некоторое значение второго столбца таблицы 1), N – количество однотипных БИД/ЗИП.

После применения всех вышеописанных операций получаем вектор диапазонов необходимого запаса для БИД/ЗИП в зависимости от времени. Значения этого вектора могут позволить эффективнее регулировать количество запасов БИД/ЗИП и планово-предупредительные работы.

Приведём блок-схему алгоритма описанного принципа работы интеллектуальной системы:

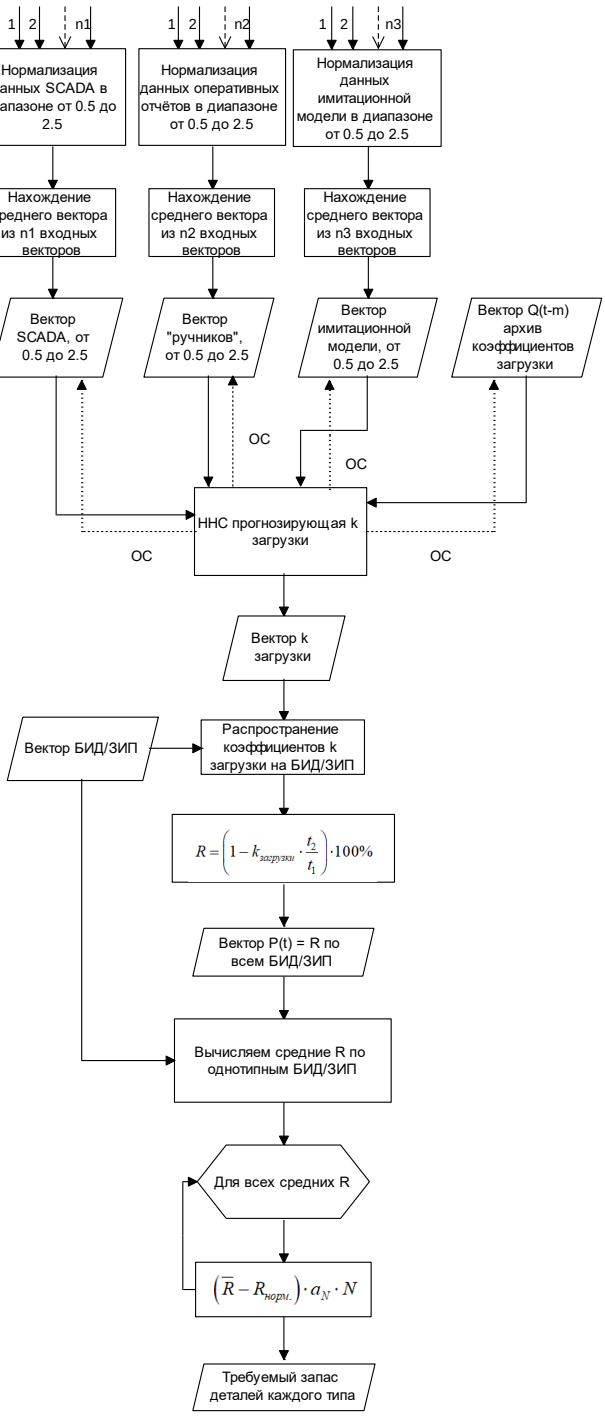


Рисунок 1.2 – Блок-схема алгоритма вычисления остаточного ресурса статистическим методом

Таким образом, алгоритм вычисления остаточного ресурса статистическим методом заключается в следующем:

1. Данные об одном объекте, полученные датчиками SCADA, по оперативным отчётам и имитационной моделью нормируются в

интервале от 0,5 до 2,5. Далее, каждая из трёх матриц (могут иметь произвольные размеры) приводится к среднему вектору.

2. После выполнения первого пункта получены три вектора: вектор SCADA, оперативных отчётов и имитационной модели. Эти векторы и вектор $Q(t-m)$ – архив коэффициентов загрузки (нужен для обучения) образуют таблицу обучения, которая передаётся на вход ННС. После обучения ННС прогнозирует коэффициент $k_{загрузки}$ по первым трём столбцам и имеет обратную связь со сформированной таблицей, чтобы в последствии можно было использовать эти данные для нового обучения ННС. Вид исходной таблицы обучения представлен в Приложении А.
3. Результатом работы ННС является вектор коэффициентов $k_{загрузки}$ (вектор от времени). Полученный вектор распространяется на БИД/ЗИП при условии, что все входящие в объект БИД/ЗИП имеют такой же коэффициент загрузки, как и сам объект. Таким образом, получаем матрицу, в которой число строк совпадает с числом БИД/ЗИП входящих в объект, и каждая строка является вектором коэффициентов $k_{загрузки}$.
4. После получения вектора коэффициентов $k_{загрузки}$, по формуле (1.6) вычисляется остаточный ресурс R каждого БИД/ЗИП. В данном случае t_2 – нормативное время эксплуатации для конкретного БИД/ЗИП, t_1 – текущее время эксплуатации этого БИД/ЗИП. На выходе для каждого БИД/ЗИП получаем вектор $P(t)=R$ зависящий от времени.
5. По однотипным БИД/ЗИП вычисляются средние значения остаточного ресурса $\overline{R}$.
6. Исходя из среднего значения остаточного ресурса однотипных деталей и количества установленных позиций этих однотипных деталей вычисляется необходимый запас БИД/ЗИП. Для этого используется формула

(1.7). В итоге получаем диапазоны необходимого запаса для каждого БИД/ЗИП в зависимости от времени.

1.8. Программная реализация

Реализация нейро-нечёткой сети в данной работе написана на языке программирования Python. Каждый слой нечёткой части ННС описан как отдельный класс со своими функциями инициализации и прямого прохода. Так, входной слой (*Слой 1*) реализуется с помощью класса FuzzifyLayer, *Слой 2,* формирующий правила, с помощью класса AntecedentLayer, слой нормализации (*Слой 3*) реализуется функцией Python – normalize, слою консеквентов (*Слой 4*) соответствует класс ConsequentLayer, а сумматору (*Слой 5*) – класс WeightedSumLayer. Также, необходим класс для описания нечёткой переменной – FuzzifyVariable, который аналогично содержит функции инициализации и вычисления. Кроме того, создан класс самой модели ANFIS в котором последовательно инициализируются все пять слоёв – это класс AnfisNet. Аналогично отдельный класс создан для реализации линейной части нейро-нечёткой сети – LinearNet, этот класс представляет собой структуру многослойной нейронной сети, имеет функцию инициализации, которая на вход принимает количество внутренних скрытых слоёв и нейронов на них, а также список функций активаций, и функцию, реализующую прямой проход сети. Разработаны функция train реализующая процесс обучения линейной части нейро-нечёткой сети и класс TrainedNFS – представляющий полную структуру нейро-нечёткой сети, т.е. ANFIS модель и многослойную нейронную сеть, класс также имеет функцию прямого прохода, необходимую для тестирования обученной нейро-нечёткой сети.

Далее, реализованы функции и классы, позволяющие выбирать тип функции принадлежности при обучении сети и автоматически подбирающие параметры этих функций принадлежности. Для каждого типа требуемых функций принадлежности создан описывающий их класс, т. е. классы SigmoidMembFunc, GaussMembFunc и TrapezoidalMembFunc для сигмоидальной, Гаусса и трапециевидной функций соответственно. Каждый из классов содержит

функции инициализации и вычисления соответствующей функции принадлежности. Также реализованы функции для создания набора функций принадлежности того или иного типа исходя из набора их параметров, т. е. функции make_gauss_mfs, make_trap_mfs и make_sigmoid_mfs для задания функций Гаусса, трапециевидной и сигмоидальной соответственно. На вход эти функции принимают значения соответствующих им параметров. Так для трапециевидной функции это два параметра: a, d из формулы (1.3), ширина верхнего основания трапеции и список центров верхнего основания, для функции Гаусса это параметр σ и список параметров μ из формулы (1.5), а для сигмоидальной функции это параметр a и список параметров b из формулы (1.4). Каждая из данных функций возвращает значения заданного количества соответствующих функций принадлежности в каждой точке из массива входных данных. Вышеперечисленные функции вызываются в функции make_anfis, которая принимает на вход массив данных для обучения, количество и тип функций принадлежности, количество выходных данных и другие параметры, а возвращает структуру ANFIS модели. В этой же функции генерируются входные параметры функций принадлежности в зависимости от вида исходных данных.

Чтобы улучшить наглядность процесса обучения сети, реализована возможность построения различных графиков. Для этого создано несколько соответствующих функций: plotErrors - построение графика ошибок, plotResults – построение графика сравнения спрогнозированных значений и действительных, plot_all_mfs – построение графиков всех функций принадлежности одного столбца данных. Также созданы функции, реализующие обучение и тестирование модели ANFIS: train_anfis_with и test_anfis соответственно. На вход эти функции получют модель ANFIS, данные выборки и параметр, определяющий нужно ли строить графики. Функция, реализующая обучение, дополнительно получает в качестве входных данных количество эпох обучения, параметры, определяющие наличие дополнительных линейных слоёв и число нейронов на них, оптимизатор, критерии обучения. В процессе

тестирования и обучения необходимо вычислять ошибки, для чего реализована функция calc_error.

Для того чтобы построить и обучить модель необходим правильно организованный массив данных, чтобы считывать данные в нужном формате создана функция get_data, при помощи которой можно считать нужное количество столбцов из файла формата txt и разделить их на два тензора: входные параметры и действительные значения. Реализована функция обучения – train_hybrid, на выходе эта функция даёт обученную модель. Чтобы использовать уже обученную модель достаточно считать входные данные и передать их в эту модель, этот процесс описан в функции prediction. Все вышеперечисленные функции необходимы для пользовательской настройки обучения и использования нейро-нечёткой сети, их реализации написаны, на основе библиотек PyTorch, Numpy, Matplotlib и с использованием результатов исследований, проведённых авторами источника [12]. Программный код приведён в Приложении Б.

1.9. Вычислительный эксперимент

В соответствии с представленным алгоритмом, входной набор данных ННС состоит из трёх векторов: а) вектор взвешенной суммы по всем датчикам, б) вектор взвешенной суммы по всем оперативным отчётам и в) вектор взвешенной суммы по данным имитационной модели. Размер входных данных составляет 46 077 наблюдений с интервалом в 1 секунду (Приложение А). Данные, являющиеся тестовой выборкой с некоторым коэффициентом масштабирования, предоставлены индустриальным партнером АО УК «ЭФКО».

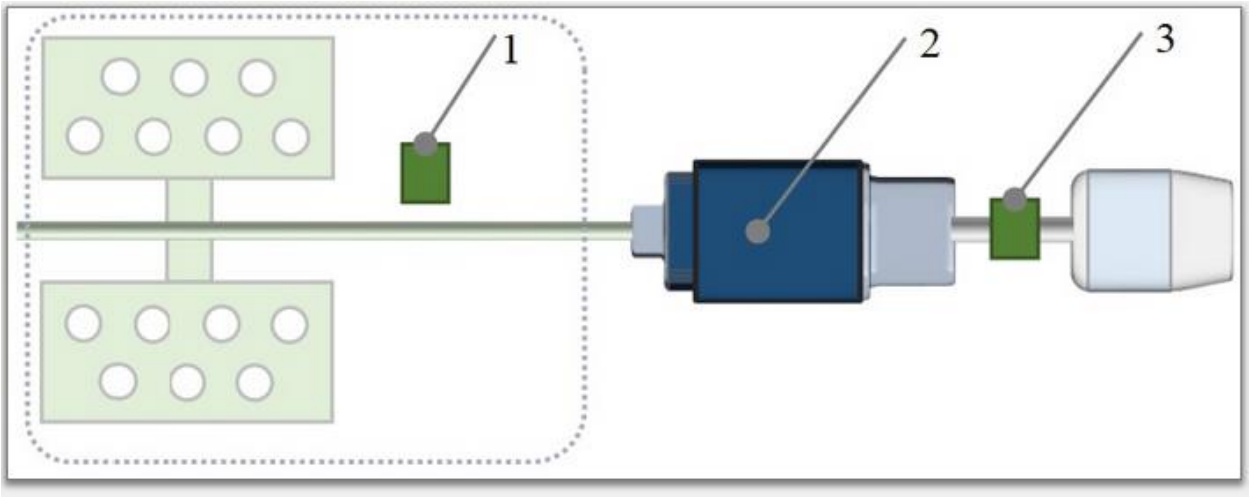
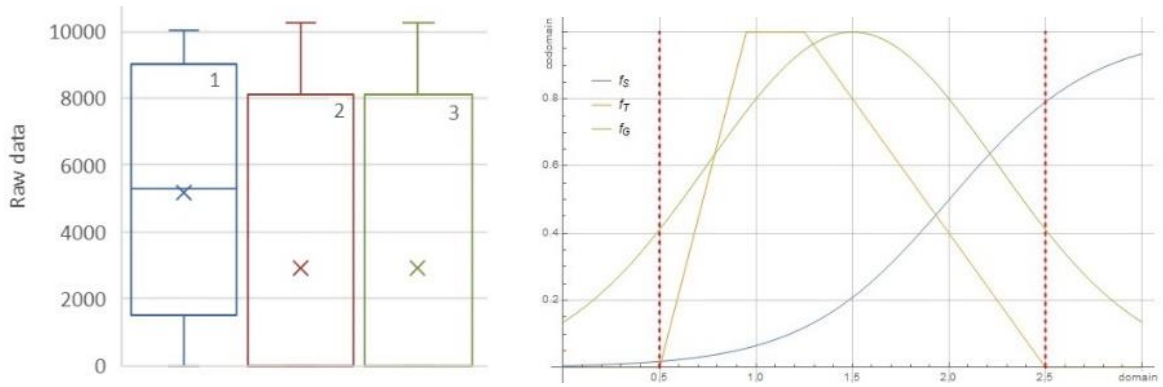


Рисунок 1.3 – Датчики электропривода мешалки реактора гидрогенизатора: 1 – датчик уровня масла, 2 – электронный и инфракрасный датчики температуры, 3 – датчик тока.

На рисунке 1.4а показаны диаграммы размаха входных необработанных данных. Можно увидеть, что для датчиков Кельвина (красный прямоугольник) и электронных датчиков IFM (зелёный прямоугольник) первый квартиль равен медиане.

В данном случае средние значения нормализованных входных данных задают область определения функций принадлежности $D = [0.5, 2.5]$, а область значений всегда представляет собой интервал $[0,1]$. В соответствии с идеей [13], мы оценили параметры функций принадлежности на нормализованном наборе данных. На рисунке 1.4б, можно увидеть области определения и значения используемых функций принадлежности (формулы): $f_S\left(x, \frac{8}{3}, 2\right)$, $f_T\left(x, 0.5, \frac{19}{20}, \frac{25}{20}, 2.5\right)$, $f_G\left(x, 1.5, 0.75\right)$.



а) б)

Рисунок 1.4 – а) Диаграммы размаха входных данных, б) Области определения и значения функций принадлежности для входных данных

В процессе выполнения работы было проведено обучение и тестирование нейро-нечёткой сети на обучающей и тестовой выборках соответственно (приведены в Приложении А). Опишем полученные результаты.

Для вычислительных экспериментов была создана сеть со следующими параметрами: 3 входных нейрона с 3 функциями принадлежности на каждом, для первого задана функция Гаусса в качестве функции принадлежности, для остальных – сигмоидальные функции; линейная часть имеет 3 слоя с 10 нейронами на каждом и использует сигмоидальные функции активации; обучение происходит за 400 эпох; в качестве оптимизатора используется устойчивый алгоритм обратного распространения (Rprop). Структура сети приведена в Приложении В.

Опишем процесс подбора настраиваемых параметров. Тип функций принадлежности для каждого входного нейрона ННС выбирается пользователем. Так как указанные функции имеют различные формы, это даёт возможность описывать разные системы оценивания, для различных наборов входных данных. Например, кусочно-линейные функции принадлежности хорошо подходят для данных с ограниченным диапазоном возможных значений и имеющих простые линейные зависимости. В нашем случае несмотря на то, что данные обучающей выборки ограничены диапазоном $[0, 2.5]$, а стандартные входные данные будут находиться в диапазоне $[0.5, 2.5]$, ННС должна

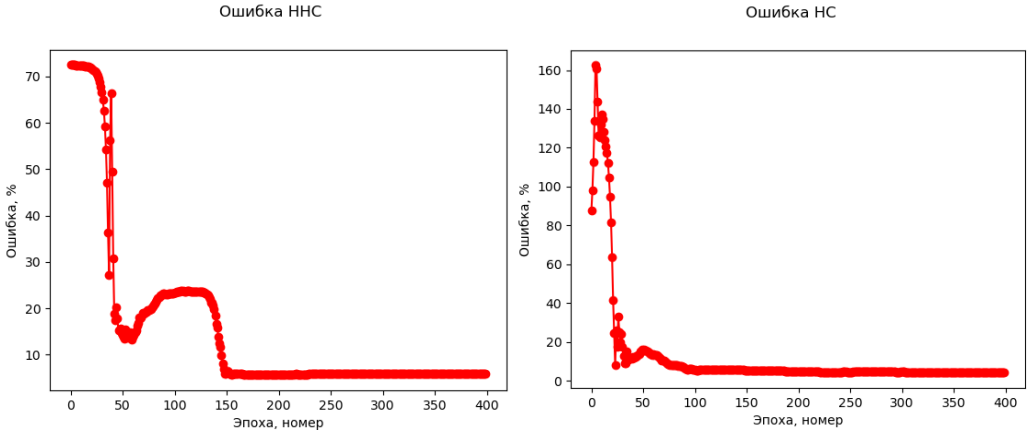
показывать адекватные результаты и вне этого диапазона, при экстремальных загрузках оборудования. Другой вариант – П-образные функции принадлежности, реализованные в данной работе при помощи функции Гаусса, так же лучше подходят для ограниченных данных, но в некоторых случаях могут хорошо описывать данные ограниченные с одной стороны, а также хорошо подходят для данных имеющих более сложные, нелинейные зависимости. Наконец, можно использовать сигмоидальную функцию принадлежности, которая подходит для описания неограниченных данных. Таким образом, для рассматриваемых данных, лучше всего использовать сигмоидальные функции принадлежности, но также можно пробовать применять функции Гаусса. По результатам экспериментов, для первого входного нейрона, соответствующего вектору данным с системы датчиков, лучшие результаты сеть показала при использовании функции Гаусса, в качестве функций принадлежности, а для остальных нейронов – при использовании сигмоидальных функций.

Кроме типа функций следует выбрать количество функций принадлежности, здесь нужно отталкиваться от того, что в процессе обучения, различные функции принадлежности, построенные для одного нейрона, должны иметь достаточно разнообразный вид, чтобы обеспечить большую вариативность правил, и всего функций принадлежности должно быть не меньше 2. В данном случае, оптимальным количеством является 3 функции принадлежности, так как при большем числе появляются слишком похожие функции.

Для линейной части сети необходимо определить количество слоёв, нейронов на них и функции активации. При определении последних можно оперировать теми же рассуждениями, что и при определении функций активации, но следует учитывать, что на вход линейной части поступает уже только один вектор, в рассматриваемом случае сигмоидальная функция активации (реализация Sigmoid из библиотеки PyTorch) ожидаемо дала лучшие результаты. Определить количество слоёв и нейронов на них можно последовательным перебором учитывая, что чем больше слоёв и нейронов, тем

точнее результаты и больше время обучения, но следует помнить, что при данной конфигурации сети слоёв должно быть не меньше двух. Таким образом определено, что сеть с 3 слоями и 10 нейронами на скрытом слое даёт наилучшие результаты. Также следует выбрать количество эпох обучения, что тоже можно сделать перебором, учитывая, что при обучении нечёткой части обычно возникает порог обучаемости ниже которого значение ошибки не падает, а также могут возникать скачки ошибки обучения. Следуя таким рассуждениям, было выбрано 400 эпох обучения.

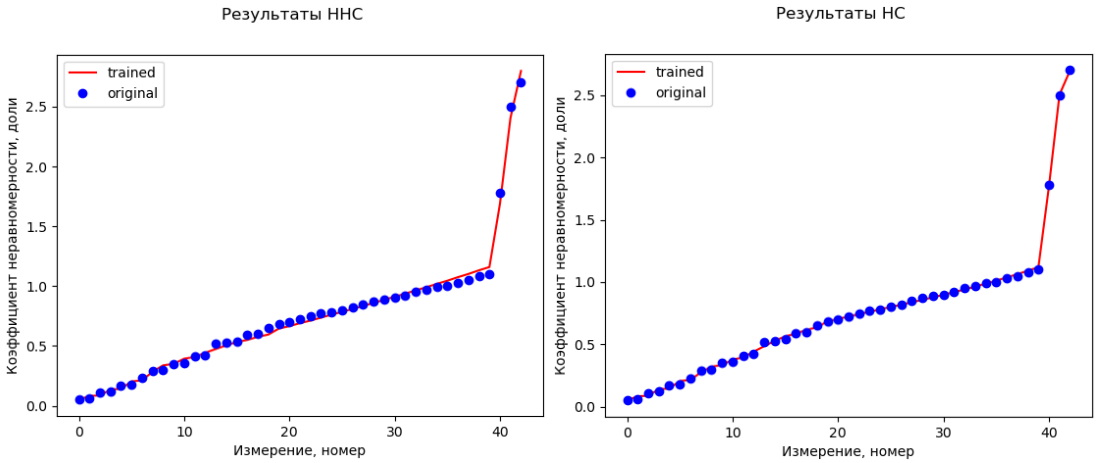
По результатам обучения приведены графики сходимости ошибок нечёткой и линейной частей ННС (рис. 1.5а, 1.5б), итоговая ошибка не превышает 5 %, при условии, что ошибка нечёткой части ННС не превышает 6%.



а) б)

Рисунок 1.5 – Графики ошибок обучения для а) ANFIS части нейро-нечёткой сети (5,88%), б) линейной части нейро-нечёткой сети (4,14%)

Также приведем графики сопоставления спрогнозированных и действительных значений по обучающей выборке по результатам обучения нечёткой и линейной частей ННС (рис. 1.6а, 1.6б).



а) б)

Рисунок 1.6 – Сравнительные графики спрогнозированных значений с реальными данными для а) ANFIS части нейро-нечёткой сети, б) линейной части нейро-нечёткой сети

Приведем графики полученных в результате обучения функций принадлежности (рис. 1.7-1.9).

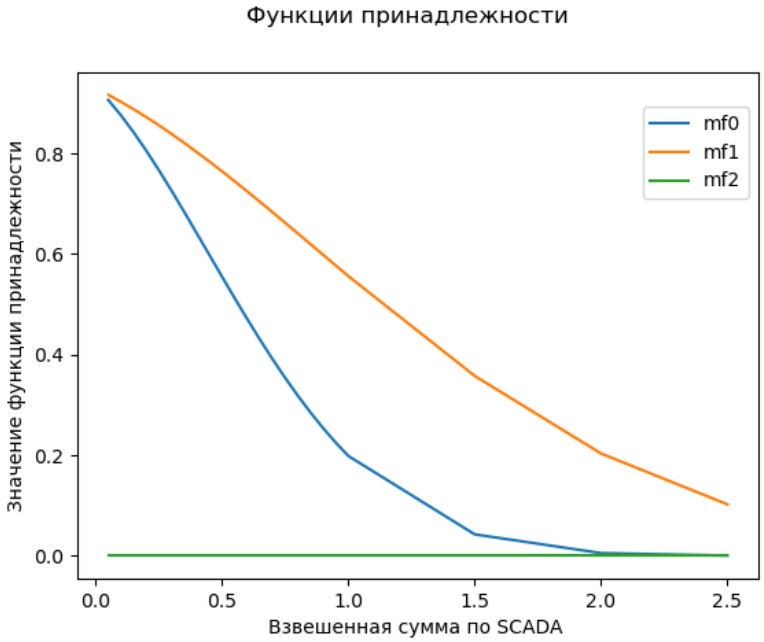


Рисунок 1.7 – График функций принадлежности для первого входного нейрона: mf0 – $f_G(x,-0.26,0.70)$, mf1 – $f_G(x,-0.54,1.43)$, mf2 – $f_G(x,3.68,-0.06)$

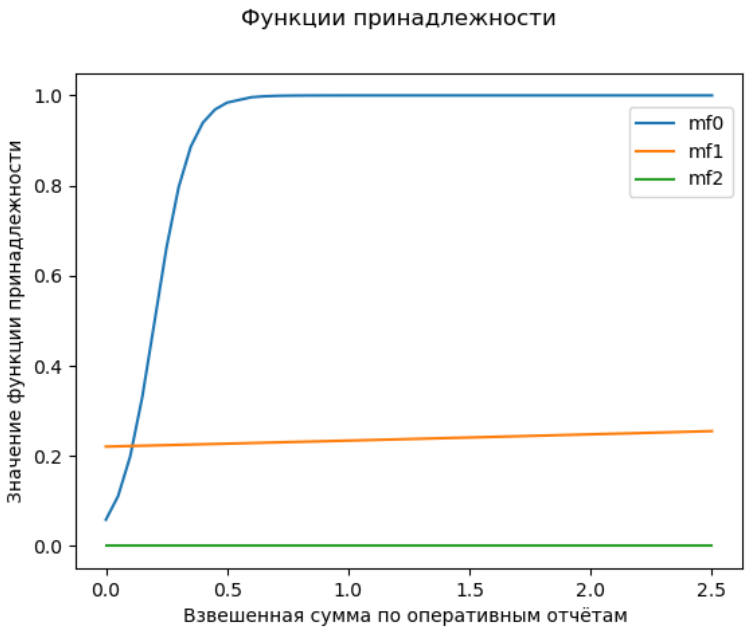


Рисунок 1.8 – График функций принадлежности для второго входного нейрона: mf0 – $f_S(x,13.80,0.20)$, mf1 – $f_S(x,0.08,16.59)$, mf2 – $f_S(x,5.12,13.82)$

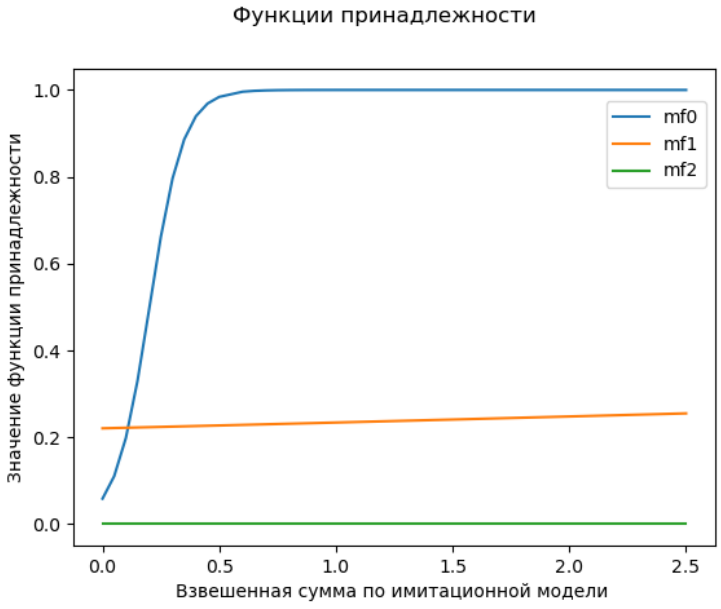


Рисунок 1.9 – График функций принадлежности для третьего входного нейрона: mf0 – $f_S(x,13.80,0.20)$, mf1 – $f_S(x,0.08,16.59)$, mf2 – $f_S(x,5.12,13.82)$

Приведём результаты тестирования. Для их оценки проведём некоторые вычисления. Найдём среднеквадратичную ошибку – средний квадрат разности между спрогнозированной величиной и действительной:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - Y_i\right)^2 \tag{1.8}$$

где n – величина выборки, Y_i – спрогнозированная величина, Y_i – действительное значение в той же точке.

Можно привести вычисление ошибки RMS, которая представляет собой корень из среднеквадратичной ошибки.

$$RMSE = \sqrt{MSE} \tag{1.9}$$

Для наглядной оценки результатов будем вычислять ошибку обучения в процентах:

$$PE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{Y_i - Y_i}{Y_i}\right| \tag{1.10}$$

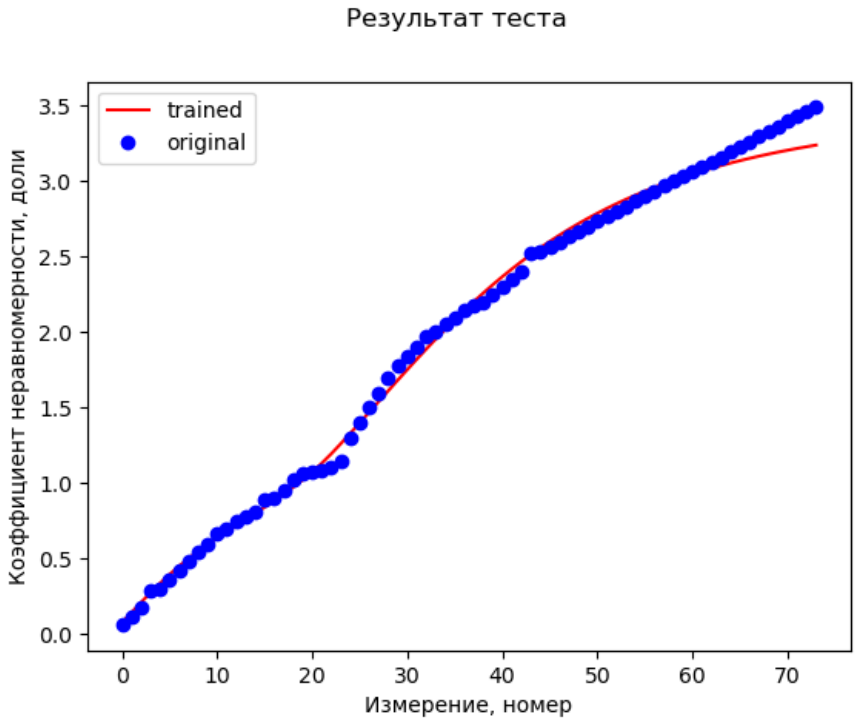


Рисунок 1.10 – Сравнительный график спрогнозированных при тестировании значений с реальными данными

Значения ошибок составили:

$$MSE = 0.00581, RMSE = 0.07623, PE = 3.90\%.$$

1.10. Выводы

Разработана интеллектуальная система на основе адаптивной нейро-нечёткой системы вывода. Созданная ННС имеет возможность пользовательской настройки параметров: вид и количество функций принадлежности (Гаусса, сигмоидальная, трапециевидная), количество линейных слоёв и нейронов на них, количество эпох обучения, количество входных нейронов, вид функций активации. На тестовом наборе данных, содержащем 3 признака и 74 наблюдения, при использовании функции Гаусса и двух сигмоидальных функций в качестве функций принадлежности и алгоритмом устойчивого обратного распространения со скоростью обучения 0,0001, за 400 эпох обучения достигается ошибка прогнозирования на тестовой выборке в 3,9%, что удовлетворяет требуемому уровню ошибки (3-5%).

Разработанная структура нейро-нечёткой сети позволяет использовать алгоритмы обучения нейронных сетей, но в то же время позволяет создавать пользовательские правила и использовать их в процессе прогнозирования.

Реализованная модель предназначена для прогнозирования коэффициента неравномерности использования производственного оборудования, на основе данных полученных из системы SCADA, по результатам «ручных» измерений и в результате работы имитационной модели. Исходя из значения спрогнозированного коэффициента вычисляется остаточный ресурс оборудования, который используется для планирования ремонтных работ и закупок запасных элементов.

2. Модуль формирования оптимального графика планово-предупредительных ремонтных работ

2.1. Описание проблемы

Существенным недостатком системы ППР является рост эксплуатационных расходов предприятия. Для повышения экономической эффективности метода необходимо учитывать остаточную стоимость заменяемого оборудования и стоимость простоя элемента останова. В качестве элемента останова может выступать цех или секция – уровень производства, которое прекращает работу на время ремонта. Под стоимостью простоя элемента останова подразумевается объём выручки, который компания теряет из-за остановки производства. Остаточная стоимость оборудования – прогнозируемое время работы детали до полного износа в денежном эквиваленте. Высокая остаточная стоимость заменяемых деталей приводит к увеличению объёма закупок.

Задача раздела – разработать алгоритм принятия решений, позволяющий сформировать оптимальный график планово-предупредительных работ. Критерий оптимальности – минимальная сумма остаточной стоимости оборудования и стоимости простоя элемента останова при заданных датах ППР и информации об оборудовании на производстве.

2.2. Подготовка данных

В рамках данной работы допустим, что замены производятся на самом низком уровне технических единиц – детали функциональных элементов оборудования, которые не имеют составных частей. Для реализации алгоритма принятия решений необходимы следующие данные о заменяемых деталях:

1. Коэффициент нагрузки;
2. Нормативное время эксплуатации;
3. Данные об осуществлении замены детали в комплекте с другими элементами агрегата или в единичном экземпляре;
4. Функциональный элемент, в состав которого входит деталь;
5. Продолжительность ремонта при одиночной замене детали;

6. Продолжительность ремонта при замене детали вместе с другими составляющими функционального элемента;
7. Стоимость детали без учёта инфляции;
8. Предельно допустимое отклонение от даты выработки ресурса.

Коэффициент нагрузки и нормативное время эксплуатации используются в расчёте остаточного ресурса детали на момент формирования графика ППР и прогнозирования остаточного ресурса на момент даты проведения ППР по формуле (1.6).

2.3. Формальная постановка задачи

Введём следующие обозначения:

N – множество всех деталей секции производственной площадки, $|N| = n$.

Элементы множества N имеют следующие характеристики:

1. $N_{T_{critical}} \in R$ – количество часов до прогнозной выработки ресурса;
2. $N_{T_{current}} \in R_+$ – текущая наработка ресурса (в часах);
3. $N_{status} \in \{0;1\}$ – индикатор, указывающий на принадлежность детали сборке;
4. $N_{T_{group}} \in R_+$ – время ремонта детали вместе с другими деталями агрегата;
5. $N_{T_{alone}} \in R_+$ – время одиночного ремонта детали, причём $N_{T_{alone}} > N_{T_{group}}$;
6. $N_P \in R_+$ – стоимость детали без учёта инфляции;
7. $N_\Delta \in R_+$ – допустимая задержка замены детали (в часах).

Пусть $M \in R_+$ – множество часов от текущего времени до плановой даты ремонта, $|M| = m$; $A = (N_{i_{T_{critical}}} - M_j)_{i,j=1}^{n,m} \in R$ – матрица остаточного ресурса на момент ремонта; $B = (N_{i_{T_{critical}}} - N_{i_{T_{current}}})_i^n \in R_+$ – вектор полного ресурса; $K = \left(\frac{A_{ij}}{B_i} \right)_{i,j=1}^{n,m} \in R$ – матрица остаточного ресурса (в процентах),

$z = \{z_i \mid K_{iz_i} = \min(|K_i|), i = \overline{1,n}\} \in N$ – вектор индексов столбцов с минимальным остаточным ресурсом, $I = (\delta_{z_i j})_{i,j=1}^{n,m}$ – матрица бинарного отношения на $N \times M, I_{ij} \in \{0;1\}$, $C = (N_{i_{status}})_{i=1}^{n}$ – вектор-индикатор режима ремонта, где 0 – одиночный, 1 – групповой, $T = (I_{ij} \cdot C_i \cdot N_{i_{T_{group}}})_{i,j=1}^{n,m} + (I_{ij} \cdot (1 - C_i) \cdot N_{i_{T_{alone}}})_{i,j=1}^{n,m} \in R$ – матрица простоя оборудования (в часах), $T_{\lim} \in R_+$ – верхний порог суммарного простоя оборудования за один ремонт (в часах), $k_d \in R_+$ – коэффициент дефляции, соответствующий году ремонта, $X = k_d (K \circ I) \cdot (N_{i_p})^T \in R, i = \overline{1,n}$ – матрица остаточной стоимости детали, $\circ$ – поэлементное произведение матриц; $P_{downtime} \in R_+$ – стоимость часа простоя оборудования.

Тогда задача может быть поставлена следующим образом:

$$F = \sum_{i=1}^{n} X_i + P_{downtime} \sum_{i=1}^{n} \sum_{j=1}^{m} T_{ij} \to \min \tag{2.1}$$

$$\begin{cases} A_{ij} > N_{\Delta} & (2.2) \\ \sum_{i=1}^{n} T_{ij} \le T_{\lim}, \forall j \in [1;m] & (2.3) \end{cases}$$

Целевая функция (1.1) означает, что требуется назначить дату замены деталей оборудования с наименьшей суммой остаточной стоимости всех деталей за период планирования и стоимости простоя элемента останова (секция/цех).

Стоимость простоя элемента останова вычисляется по следующей формуле:

$$F_1 = \frac{V_{output} P_{product}}{T_{operation}} \sum_{i=1}^{n} \sum_{j=1}^{m} T_{ij} \tag{2.4}$$

где F_1 – стоимость простоя элемента останова, V_{output} – объём выпущенной продукции, $P_{product}$ – стоимость единицы продукции, $T_{operation}$ – время работы за период в часах.

Стоимость простоя элемента останова вычисляется по следующей формуле:

$$F_2 = \sum_{i=1}^{n} X_i \qquad (2.5)$$

где X_i – остаточная стоимость *i*-й детали.

Далее введем операторы: $W_{ij}[I] = I \circ (1 - \delta_{ki}\delta_{lj})_{k,l=1}^{n,m} + (\delta_{ki}\delta_{lj-1})_{i,j=1}^{n,m}$ – оператор, сдвигающий дату ремонта *i*-ой детали на предыдущую дату. Область определения оператора W – множество матриц I, для которых $\exists I_{kj-1} = 1, k = \overline{1,n}$ такое, что $N_i \in U_{N_i}, N_k \in U_{N_i}, U_{N_i} \subseteq N$, здесь U – подмножество деталей одного агрегата. Оператор $V_i[C] = (\delta_{ii})_{i=1}^{n} - C$ – оператор смены режима ремонта для *i*-ой детали с одиночного на групповой.

2.4. Математический алгоритм решения задачи

Поставленную задачу (2.1)-(2.3) можно отнести к классу комбинаторных задач, подобных «задаче об упаковке в контейнеры» и «задаче о рюкзаке», которые могут быть решены аналитически или асимптотически [14, 15]. Ключевое отличие поставленной задачи от упомянутых выше заключается в особенностях качественных критериев, а также изменении характеристик объектов в зависимости от варианта размещения. В рамках данной работы мы предлагаем следующий алгоритм решения задачи (2.1)-(2.3):

1. Сформировать матрицу *I:* строки соответствуют *деталям* оборудования, а столбцы – *датам* ППР. Замена детали производится в дату ППР с наименьшим по модулю значением остаточного ресурса.
2. Сформировать матрицу *A*: строки соответствуют *деталям* оборудования, а столбцы – *датам* ППР.
3. Если условие (2.2) не выполнено, то применить операторы *W* и *V* для элементов матрицы *I*. Повторить шаги 2-3. Если условие (2.2) выполнено, то на шаг 4.
4. Сформировать матрицу T: строки соответствуют *деталям* оборудования, а столбцы – *датам* ППР.

5. Если условие (2.3) верно, то на шаг 6, иначе применить операторы *W* и *V* к матрице *I*: для столбцов матрицы *T* найти минимальный элемент в соответствующем столбце матрицы *A*. Повторить шаги 2-5.
6. Сформировать матрицу *X*: строки соответствуют деталям оборудования, а столбцы – датам ППР.
7. Вычислить значение целевой функции *F* (2.1).
8. Применить оператор *W* к матрице *I*: производящего сдвиг замены *i*-й детали на предыдущую дату ППР в том случае, если в предыдущем плане есть детали из такого же агрегата, что и *i*-я деталь. Для *i*-й детали применить оператор *V* в том случае, если в первоначальном плане режим ремонта был одиночный.
9. Повторить шаги 2-7.
10. Если значение *F* уменьшилось, то оптимальным решением считается полученная матрица *I*, в противном случае – исходная матрица *I*.

2.5. Программный алгоритм решения задачи

Рассматриваемая комбинаторная задача может быть решена с помощью динамического программирования. Выделим следующие этапы алгоритма:

1. Формирование списка деталей, подлежащих замене

При составлении графика ППР указываются даты начала и завершения периода планирования, а также строится план ППР на заданный период на основании данных нормативов планирования видов ремонтов.

На данном этапе проводится сравнение прогнозной даты выработки ресурса деталей с началом и концом периода. Если прогнозная дата выработки ресурса детали меньше или равна дате конца периода и больше или равна дате начала периода, то деталь добавляется в список деталей, подлежащих ППР за период. Если прогнозная дата выработки ресурса детали меньше начала периода и удовлетворяет условию (2.3), то деталь также добавляется в список, иначе ЛПР принимает решение о внеплановой замене.

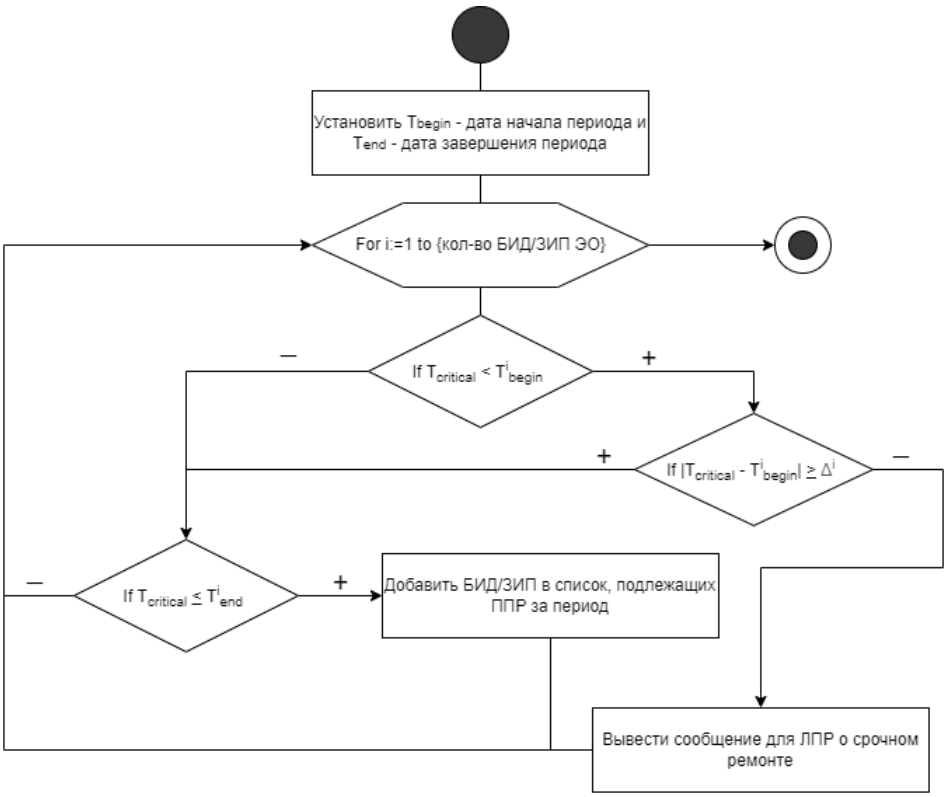


Рисунок 2.1 – Блок-схема алгоритма формирования списка деталей

2. *Формирование базового плана по остаточному ресурсу оборудования*

Поскольку предполагается, что запрос на расчёт нового плана ППР может быть осуществлён в любой момент рассматриваемого периода, при каждом новом распределении деталей необходимо определить дату начала ближайшего будущего ППР. Для каждой детали из сформированного списка назначается дата начала ППР по следующему алгоритму:

1. Определяется ближайшая по модулю дата ППР;
2. Если время выработки ресурса рассматриваемой детали больше, чем дата ближайшего к нему ППР, то вносим его в список на ближайший к нему ППР;
3. Иначе, если разница между датой ближайшего к нему ППР и датой выработки ресурса не превышает предельно допустимое отклонение для данной категории, то вносим деталь в список на ближайший к нему ППР;
4. Иначе вносим деталь в список на предыдущий от ближайшего к нему ППР.

Результатом этапа будет двумерная матрица, в которой строки – перечень деталей, входящих в план ППР, а столбцы – даты запланированных ППР, начиная с даты формирования запроса на расчёт. В ячейках на данном этапе содержатся значения: 1, если замена детали планируется в соответствующую дату ППР; 0, если замена детали не планируется в данную дату ППР.

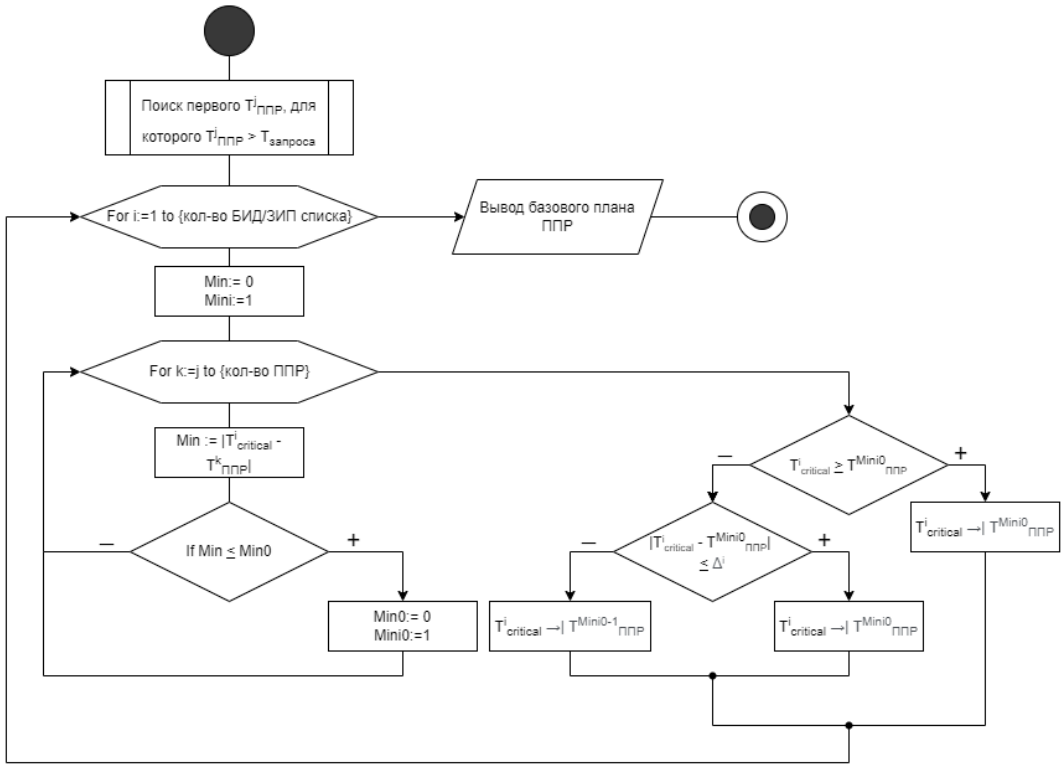


Рисунок 2.2 – Блок-схема алгоритма формирования базового плана

3. Заполнение плана значениями остаточного ресурса оборудования

Для решения задачи необходимы данные об осуществлении замены детали в комплекте с другими элементами агрегата или в единичном экземпляре. Для каждой детали известны время замены в комплекте (t_{group}) и время единичной замены (t_{alone}), причём $t_{group} < t_{alone}$. Проводится расчёт суммарного времени простоя элемента останова. В том случае, если на текущую дату ППР суммарное время простоя превышает допустимое значение, то деталь с наименьшим остаточным ресурсом на момент запланированной даты ППР переносится в список на предыдущую дату ППР.

Далее производится расчёт суммарной стоимости простоя элемента останова и суммарной остаточной стоимости деталей на планируемые даты ППР

по формулам (2.4) и (2.5). На основании полученных значений вычисляется целевая функция (2.1).

Результатом этапа будет двумерная матрица, в которой строки соответствуют перечню юнитов, входящих в план ППР, а столбцы – датам запланированных ППР, начиная с даты формирования запроса на расчёт. В ячейках матрицы содержатся значения времени замены деталей. Полученный план считается оптимальным по критерию F_2.

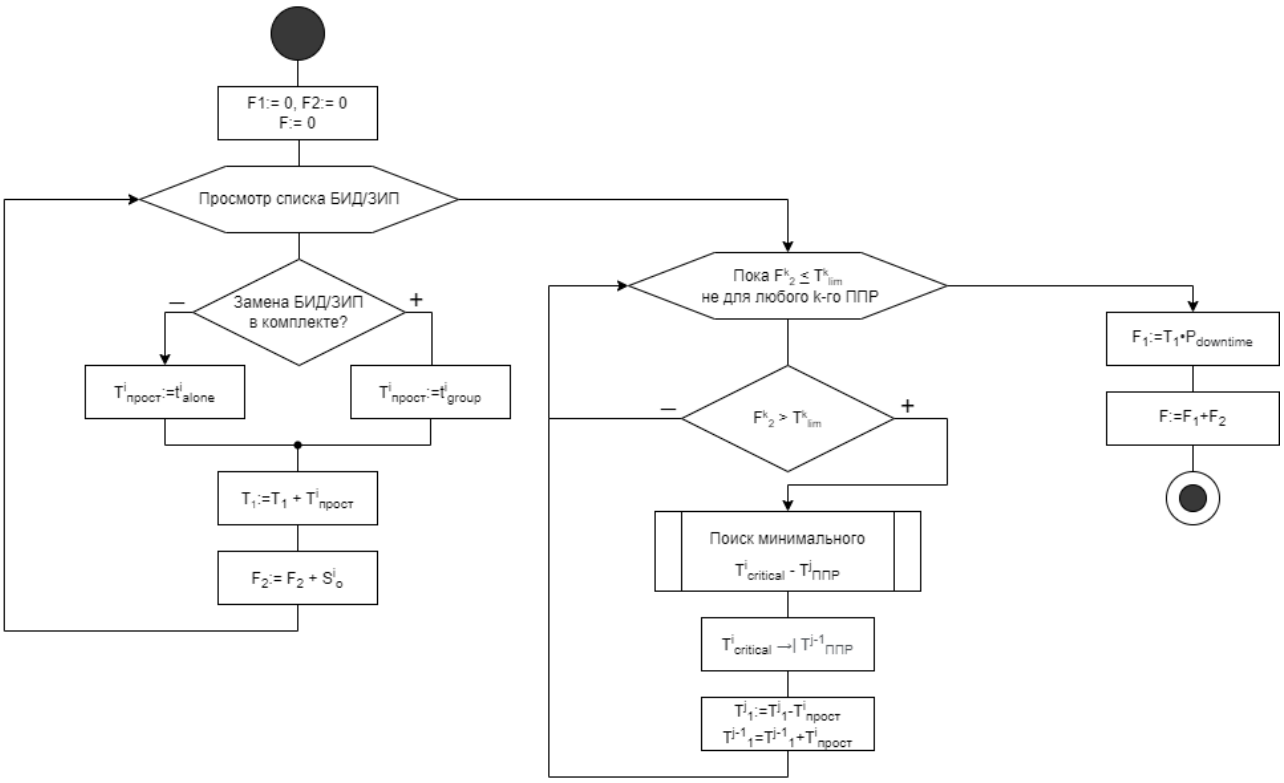


Рисунок 2.3 – Блок-схема алгоритма заполнения плана значениями остаточного ресурса

4. Оптимизация распределения деталей

Если на замену детали тратится время его единичной замены, то необходимо проверить, есть ли в списке на предыдущую дату ППР детали из этого агрегата. Если они есть, то замена детали переносится в список на предыдущую дату ППР, занося срок замены в комплекте. На каждом этапе проводится расчёт целевой функции F для плана ППР при условии замены на предыдущую дату. Если значение F уменьшилось, то сохраняем обновлённым

план, иначе – отменяем перенос детали в список предыдущего ППР. Данная процедура повторяется до конца матрицы.

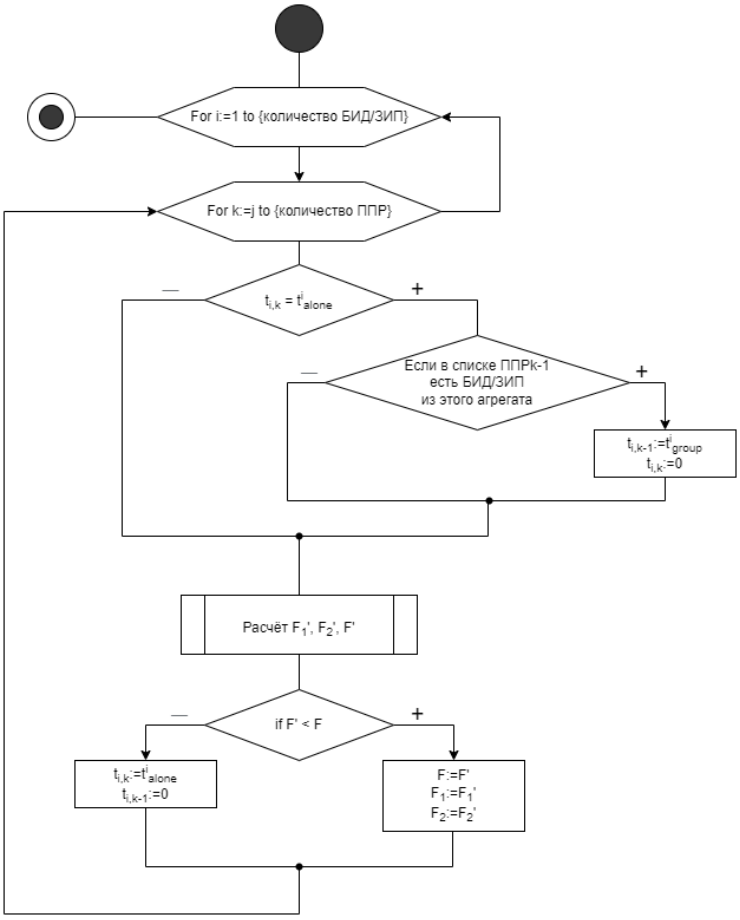


Рисунок 2.4 – Блок-схема алгоритма формирования оптимального плана

Программный код реализации алгоритма формирования оптимального графика ППР приведён в Приложении Г.

3. Проектирование системы

3.1. Структура информационной системы

Разрабатываемая информационная система состоит из нескольких блоков: «Авторизация», «База данных», «Анализ экономических показателей» и «Имитационные модели и ННС». Определим структуру каждого блока.

Окно общего вида блока «База данных» включает в себя таблицу с содержимым базы данных и необходимой сводной информацией, а также следующие вкладки:

1. Оценка остаточного измерения;
2. Замены;

3. Телеизмерения;
4. Ручные замеры;
5. Имитационные расчёты.

Данный блок включает в себя диалоговое окно «Прогноз на дату» для вывода информации по остаточному ресурсу за установленный промежуток времени.

Блок также включает диалоговое окно «Вывод графиков» для отображения графиков состояния объектов на заданном уровне иерархии технологических единиц.

Блок «Анализ экономических показателей» содержит четыре вкладки: «Экономия ресурсов от своевременной замены», «Текущая стоимость БИД/ЗИП» и «Стоимость избыточного запаса БИД/ЗИП».

Данный блок включает в себя диалоговое окно «Экономия ресурсов» с графиками, отображающими экономию ресурсов с использованием алгоритма оптимального распределения деталей.

3.2. Формирование требований к разрабатываемой системе

3.2.1. Блок «Авторизация»

Окно авторизации должно включать следующий функционал:

1. Возможность ввода логина пользователя;
2. Возможность ввода пароля пользователя;
3. Проверка существования пользователя в системе;
4. Проверка пароля;
5. Обеспечен переход на окно регистрации.

В окне регистрации требуется разработать следующий функционал:

1. Ввод логина пользователя (обязательное поле);
2. Ввод фамилии, имени (обязательные поля) и отчества (необязательное поле)
3. Выбор должности из перечня предложенных (обязательное поле);
4. Выбор уровня доступа (администратор, пользователь 1-го уровня, пользователь 2-го уровня и т.п.) (обязательное поле);

5. Ввод пароля (минимум 8 символов);
6. Подтверждение пароля;
7. Проверка уникальности логина;
8. Требование подтверждения администратором при регистрации пользователя с уровнем доступа «Администратор»;
9. Проверка пароля.

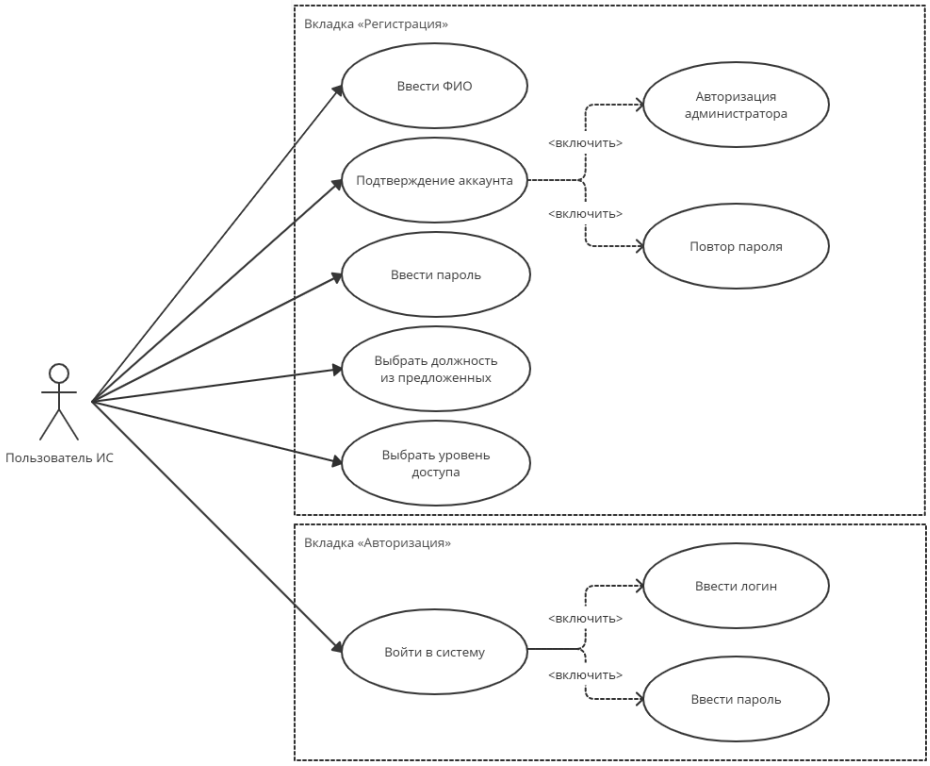


Рисунок 3.1 – Диаграмма прецедентов блока «Авторизация»

3.2.2. Блок «База данных»

В окне общего вида должен быть разработан следующий функционал:

6. Вывод таблицы с содержимым базы данных на каждом уровне иерархии БД (начиная с уровня «Производственные площадки»);
7. Переключение на следующий уровень иерархии при нажатии на выбранный объект в таблице;
8. Возврат на предыдущий уровень БД;
9. Добавление объектов в БД;
10. Удаление объектов из БД.

Сводная таблица базы данных должна иметь следующее содержание:

- Для уровня «Производственные площадки»: отображение списка всех производственных площадок со следующей информацией: название, количество цехов, количество секций, количество технологических объектов, количество агрегатов, количество деталей, количество БИД/ЗИП;
- На уровне «Производственная площадка» должен выводиться список цехов на выбранной площадке со следующей информацией: название цеха, количество секций, количество технологических объектов, количество агрегатов, количество деталей, количество БИД/ЗИП, остаточный ресурс в процентах, остаточный ресурс в часах;
- На уровне «Цех» должен выводиться список секций данного цеха со следующей информацией: название секции, количество технологических объектов, количество агрегатов, количество деталей, количество БИД/ЗИП, остаточный ресурс в процентах, остаточный ресурс в часах;
- На уровне «Секция» должен выводиться список технологических объектов данной секции со следующей информацией: технологический код объекта, название объекта, количество агрегатов, количество деталей, количество БИД/ЗИП, остаточный ресурс в процентах, остаточный ресурс в часах;
- На уровне «Технологический объект» должен выводиться список функциональных элементов данного технологического объекта со следующей информацией: технологический код элемента, название элемента, количество деталей, количество БИД/ЗИП, код МДМ элемента, остаточный ресурс в процентах, остаточный ресурс в часах;
- На уровне «Функциональный элемент» должен выводиться список деталей данного функционального элемента со следующей информацией: название детали, количество БИД/ЗИП, код МДМ элемента, остаточный ресурс в процентах, остаточный ресурс в часах;

- На уровне «Деталь» должен выводиться список БИД/ЗИП со следующей информацией: название БИД/ЗИП, код МДМ элемента, остаточный ресурс в процентах, остаточный ресурс в часах.

Вкладка «Оценка остаточного измерения» должна иметь следующий функционал:

1. Вывод таблицы с оценкой остаточного ресурса всех элементов, деталей и БИД/ЗИП, расположенных на текущем уровне БД. Таблица должна содержать следующую информацию: название цеха, на котором расположен объект; название секции, на которой расположен объект; код технологического объекта, на котором расположен элемент, деталь или БИД/ЗИП; код функционального элемента; название детали, если у функционального элемента есть детали; название БИД/ЗИП, если деталь делима на БИД/ЗИП; дата установки объекта; приоритет; остаточный ресурс в процентах; остаточный ресурс в часах; прогнозная дата выработки ресурса; оптимальная дата начала ремонта.
2. Сортировка и цветовая индикация элементов таблицы в соответствии с остаточным ресурсом или по оптимальной дате начала ремонта. Введём цветовые обозначения для каждого состояния объекта: красный – критическое состояние, оранжевый – плохое состояние, жёлтый – удовлетворительное состояние, светло-зелёный – хорошее состояние, зелёный – отличное состояние. Объект находится в критическом состоянии, если его остаточный ресурс менее 20%, либо до оптимальной даты начала ремонта осталось менее двух недель. Объект находится в плохом состоянии, если его остаточный ресурс менее 40%, либо до оптимальной даты начала ремонта осталось менее месяца. Объект находится в удовлетворительном состоянии, если его остаточный ресурс менее 60%, либо до оптимальной даты начала ремонта осталось менее полутора месяца. Объект находится в хорошем состоянии, если его остаточный ресурс менее 80%, либо до

оптимальной даты начала ремонта осталось менее двух месяцев. Объект находится в отличном состоянии, если его остаточный ресурс более 80%, либо до оптимальной даты начала ремонта осталось более двух месяцев.

3. Обеспечен переход на окно с выводом 3D-представления объекта;
4. Обеспечен выбор временного диапазона для отображения информации об объектах, подлежащих ремонту в данном диапазоне, в окне «Прогноз на дату»;
5. Обеспечен переход в окно «Прогноз на дату»;
6. Обеспечен переход в окно «Вывод графиков»;
7. Возможность экспорта таблицы с оценкой остаточного ресурса в файл Excel.

Вкладка «Замены» должна иметь следующий функционал:

1. Вывод таблицы замен со следующей информацией: позиция замены; код МДМ объекта; дата установки объекта; дата обнаружения дефекта; дата замены; время простоя в днях; цена объекта на момент установки в рублях; дефлятор; цена объекта на данный момент в рублях;
2. Обеспечен выбор временного диапазона для выгрузки данных, вывода графиков и экспорта данных в файл;
3. Обеспечен переход в окно с выводом табличных данных;
4. Обеспечен переход в окно вывода графиков;
5. Возможность экспорта данных в файл.

Вкладки «Телеизмерения», «Ручные замеры» и «Имитационные расчёты» должны иметь следующий функционал для значений датчиков SCADA, ручных замеров и расчётов имитационной модели соответственно:

1. Вывод таблицы объектов на данном уровне со следующей информацией: наименование объекта; описание измерения; единица измерения;
2. Обеспечен выбор временного диапазона для выгрузки данных, вывода графиков, импорта и экспорта данных в файл;

3. Обеспечен выбор объектов в таблице для вывода в отдельном окне табличных значений в установленном временном диапазоне;
4. Обеспечен переход в окно с выводом табличных данных;
5. Обеспечен переход в окно вывода графиков;
6. Возможность экспорта данных в файл;
7. Возможность импорта данных из файла.

Окно «Прогноз на дату» должен иметь следующий функционал:

1. Вывод таблицы остаточного ресурса из вкладки «Оценка остаточного измерения» с данными за указанный промежуток времени;
2. Сортировка и цветовая индикация элементов таблицы остаточного ресурса;
3. Вывод круговой диаграммы количественного соотношения состояния объектов за данный период;
4. Возможность установки новой даты прогноза и обновления таблицы;
5. Возможность выгрузки таблицы в файл Excel.

Окно «Вывод графиков» должно отображать следующую информацию:

1. График количественного распределения состояния объектов в каждом из ближайших 6 месяцев;
2. Круговая диаграмма соотношения по состоянию всех объектов на текущем уровне.

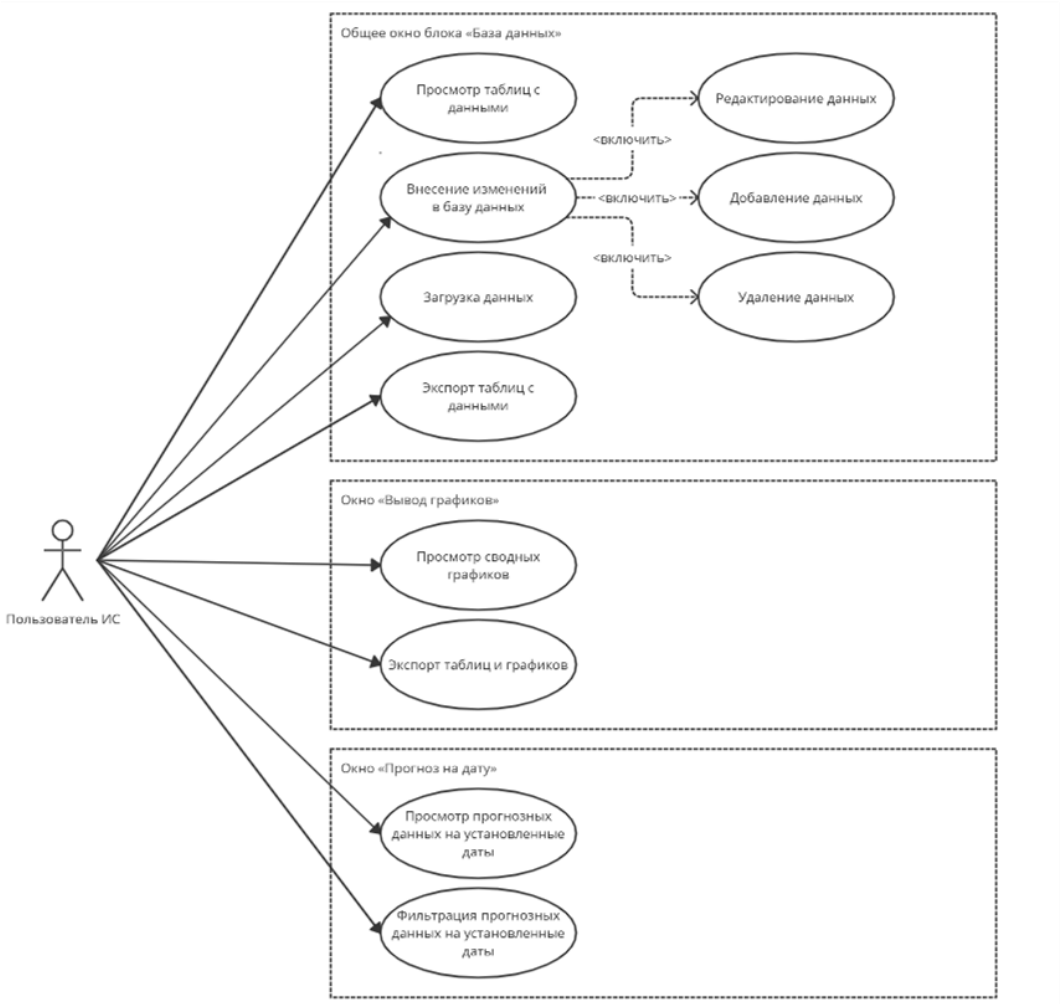


Рисунок 3.2 – Диаграмма прецедентов блока «База данных»

3.2.3. Блок «Анализ экономических показателей»

Общее окно блока «Анализ экономических показателей» должно иметь следующий функционал:

1. Возможность выбора производственной площадки, для которой формируется график ППР;
2. Возможность выбора цеха, для которого формируется график ППР;
3. Возможность выбора секции цеха;

Вкладка «Экономия ресурсов от своевременной замены» должна иметь следующий функционал:

1. Возможность установки периода планирования ППР с указанием даты начала и даты конца формирования графика;
2. Возможность ввода количества ППР за указанный период;

3. Возможность указания даты *i*-го ППР. После установки даты пользователь должен подтвердить ввод. Затем открывается возможность указания даты следующего ППР и процедура повторяется. Возможность подтверждения ввода даты ППР исчезает при указании последней даты.
4. Возможность формирования базового плана на текущую дату. Создание базового плана основывается на алгоритме формирования графика ППР без учёта критериев оптимальности – суммарной стоимости простоя оборудования и остаточной стоимости деталей.
5. Возможность оптимизации базового плана на текущую дату. Данная функция доступна пользователю только в том случае, если был сформирован базовый план. Создание оптимального плана основывается на алгоритме формирования графика ППР с учётом критериев оптимальности – суммарной стоимости простоя оборудования и остаточной стоимости деталей.
6. Обеспечен переход в окно «Экономия ресурсов». Функция доступна пользователю только в том случае, если сформирован оптимальных план.
7. Формирование таблицы базового плана, в котором столбцы – даты ППР, строки – детали. Ячейки таблицы окрашены в зелёный цвет в том случае, если *i*-й детали состоится в *j*-ю дату в соответствии с базовым планом.
8. Формирование таблицы оптимального плана, в котором столбцы – даты ППР, строки – детали. Ячейки таблицы окрашены в зелёный цвет и содержат время выполнения замены в том случае, если *i*-й детали состоится в *j*-ю дату в соответствии с оптимальным планом.
9. Возможность экспорта сформированного графика в таблицу Excel.

Окно «Экономия ресурсов» должно иметь следующий функционал:

1. Формирование столбчатой диаграммы сравнения суммарной стоимости простоя элемента останова как на каждую дату ППР, так и на весь период планирования, по базовому и оптимальному плану;
2. Формирование столбчатой диаграммы сравнения суммарной остаточной стоимости оборудования как на каждую дату ППР, так и на весь период планирования, по базовому и оптимальному плану;
3. Формирование столбчатой диаграммы сравнения показателей суммарной экономии остаточной стоимости оборудования и суммарной экономии стоимости простоя элемента остановка по базовому и оптимальному плану;
4. Формирование визуального отображения экономии ресурсов по оптимальному плану относительно стоимости просто элемента останова, остаточной стоимости оборудования и в совокупности по этим критериям;
5. Возможность выгрузки отчёта по полученным показателям в таблицу Excel.

Вкладка «Текущая стоимость БИД/ЗИП» должна иметь следующий функционал:

1. Формирование таблицы БИД/ЗИП указанного элемента останова, содержащей следующую информацию: название БИД/ЗИП; стоимость установленных БИД/ЗИП; стоимость БИД/ЗИП, находящихся на складе; стоимость БИД/ЗИП, заменённых с остаточным ресурсом;
2. Формирование круговой диаграммы, разделённой на три сектора: суммарная стоимость установленных БИД/ЗИП, суммарная стоимость БИД/ЗИП в запасе и суммарная стоимость заменённых БИД/ЗИП;
3. При нажатии на сектор диаграммы происходит фильтрация в таблице по столбцу, соответствующему выбранному сектору;
4. При нажатии на строку таблицы происходит фильтрация в круговой диаграмме в соответствии с выбранным БИД/ЗИП.

Вкладка «Стоимость избыточного запаса БИД/ЗИП» должна содержать следующий функционал:

1. Формирование таблицы с информацией о суммарной стоимости простоя элемента останова, содержащей следующую информацию: название БИД/ЗИП; название технологического объекта, на котором установлен БИД/ЗИП; количество БИД/ЗИП на складе; нормативные сроки заказа; нормативные сроки поставки; нормативные сроки установки; частота выхода из строя; стоимость избыточного запаса; количество БИД/ЗИП, хранящихся на случай аварии; избыточный запас; количество однотипных БИД/ЗИП;
2. Формирование столбчатой диаграммы стоимости избыточного запаса БИД/ЗИП либо по рассматриваемым секциям, либо по технологическим объектам;
3. Возможность выбора условия формирования столбчатой диаграммы стоимости избыточного запаса;
4. Отображение суммарной стоимости избыточного запаса БИД/ЗИП на рассматриваемом уровне производства;
5. Экспорт сформированной таблицы в файл Excel.

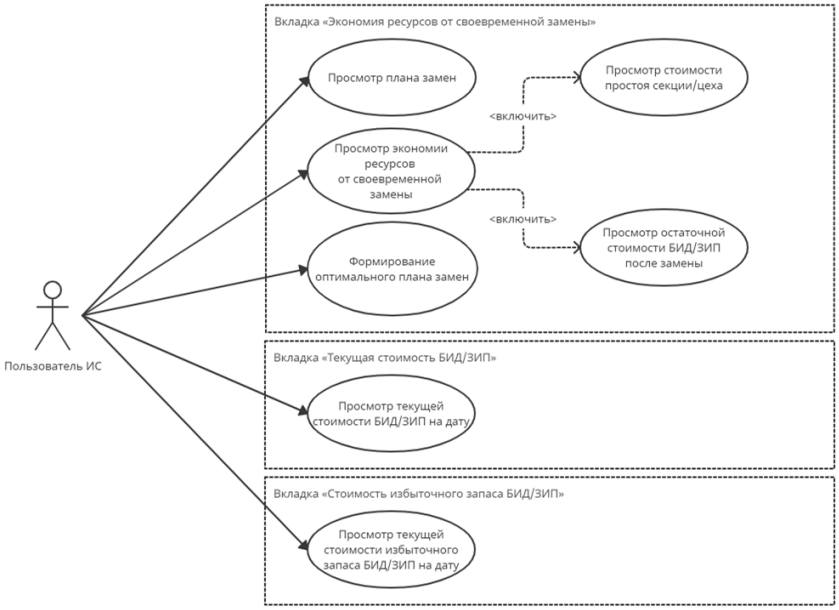


Рисунок 3.3 – Диаграмма прецедентов блока «Анализ экономических показателей»

3.2.4. Блок «Имитационные модели и ННС»

Окно блока ННС должно включать следующий функционал:

1. Выбор детали, для которой будет осуществляться расчёт остаточного ресурса из списка всех доступных. При выборе объекта должна быть доступна информация о его местоположение (секция, цех, площадка, объект), о том какая модель уже использовалась для расчёта его остаточного ресурса (путь к модели, дата создания, количество входов, точность, рассчитанный моделью остаточный ресурс).
2. Настройка количества слоёв линейной части ННС (не менее трёх) и количества нейронов на скрытых слоях (не менее одного).
3. Активация обратной связи.
4. Подробная настройка ННС, включающая в себя выбор графиков, которые необходимо отобразить по результатам обучения модели или по результатам прогноза, переход к диалоговому окну выбора функций активации по кнопке «Настройка функций активации».
5. Выбор метода обучения (гибридный или нет).
6. Настройка количества входов сети и эпох обучения (не меньше 1).
7. Настройка количества функций принадлежности (не менее двух).
8. Подробная настройка правил, включающая в себя выбор алгоритма логического вывода (Такаги-Сугено-Канга, Мамдани или Сугено), переход к выбору функций принадлежности по кнопке «Настроить функции принадлежности», переход к вводу пользовательских правил по кнопке «Ручная настройка правил».
9. Отображение расположения файла с обучающей выборкой для выбранного объекта и полученной на этапе обучения точности модели.
10. Возможность выбора файла обучающей выборки (по кнопке «Выбор файла»), создания таблицы обучения (по кнопке «Создать таблицу вручную») и обучения построенной ННС (по кнопке «Обучить ННС»).

11. Отображение расположения файла с тестовой выборкой для выбранного объекта и полученной на этапе тестирования точности модели.
12. Возможность выбора файла тестовой выборки (по кнопке «Выбор файла»), создания тестовой таблицы (по кнопке «Создать таблицу вручную») и тестирования построенной ННС (по кнопке «Протестировать ННС»).
13. Выбор метода вычисления остаточного ресурса (статистический или при помощи нейронной сети) и возможность получения остаточного ресурса для выбранных деталей.

Диалоговое окно «Функции активации» должно иметь следующий функционал:

1. Выбор типа функции активации из списка возможных: Sigmoid, GELU, ReLU.
2. Подтверждение выбора нажатием на кнопку, закрытие данного диалогового окна.

Диалоговое окно «Функции принадлежности» должно содержать следующий функционал:

1. Выбор типа функции принадлежности из списка возможных: сигмоидальная, трапециевидная, функция Гаусса.
2. Подтверждение выбора нажатием на кнопку, закрытие данного диалогового окна.

Диалоговое окно «Ручная настройка правил» должно обладать следующим функционалом:

1. Отображение количества термов (исходя из выбранного количества функций принадлежности) и возможность задания термов при нажатии на кнопку «Перейти к настройке».
2. Настройка количества следствий (не менее двух) и возможность их задания при нажатии на кнопку «Настроить следствия».

3. Настройка количества правил (не менее одного) и возможность их описания при нажатии на кнопку «Создать правила».
4. Подтверждение выбора нажатием на кнопку, закрытие данного диалогового окна.

Диалоговое окно «Создание лингвистической переменной» должно обладать следующим функционалом:

1. Ввод значения лингвистического терма.
2. Подтверждение ввода нажатием на кнопку, закрытие данного диалогового окна.

Диалоговое окно «Задание лингвистических следствий» должно обладать схожим функционалом:

1. Ввод лингвистического и численного (дробное число) значений следствия.
2. Подтверждение ввода нажатием на кнопку, закрытие данного диалогового окна.

Диалоговое окно «Создание правил» должно обладать следующим функционалом:

1. Выбор переменной условия.
2. Выбор лингвистического терма из созданных.
3. Возможность добавления условия через И/ИЛИ связь.
4. Выбор лингвистического следствия из созданных.
5. Подтверждение ввода нажатием на кнопку, закрытие данного диалогового окна.

Диалоговые окна «Создание таблицы обучения» и «Создание таблицы тестирования» должны содержать следующий функционал:

1. Выбор количества входных величин (столбцов).
2. Выбор количества значений для обучения или тестирования соответственно (строк).
3. Возможность сохранения созданной таблицы в Excel файл (с указанием класса детали и названия файла).

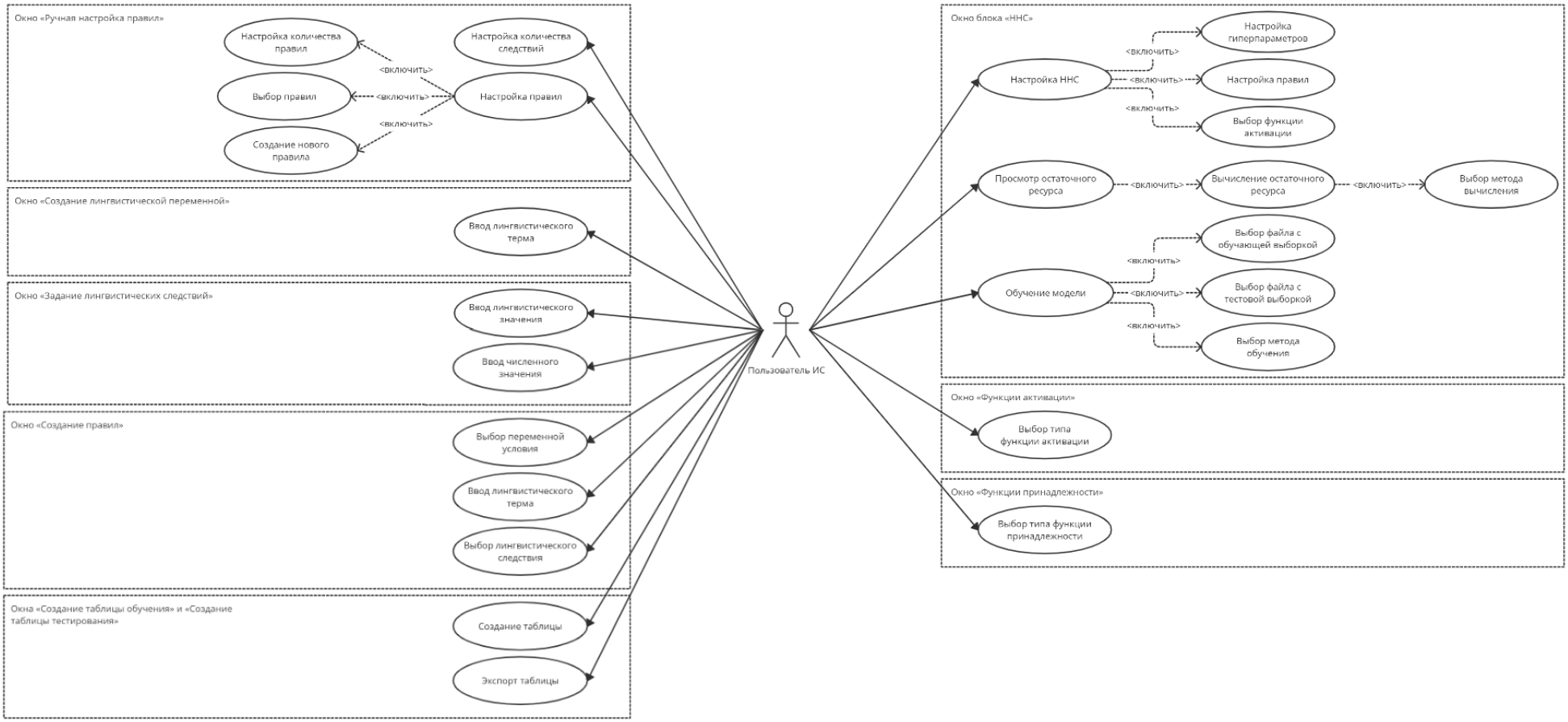


Рисунок 3.4 – Диаграмма прецедентов блока «Имитационные модели и ННС»

3.3. Архитектура системы

Для разработки программного продукта была использована монолитная схема проектирования Model-View-Controller (MVC). Данный метод разделяет архитектуру приложения на три компонента. Model (модель) – это объект приложения, содержащий экземпляры классов базы данных. View (вид) – это совокупность классов, отвечающих за экранное представление приложения. Controller (контроллер) содержит методы для выполнения запросов пользователя и манипулирования данными. Контроллер может оказывать воздействие на представление как напрямую, так и через изменение модели данных. [16]

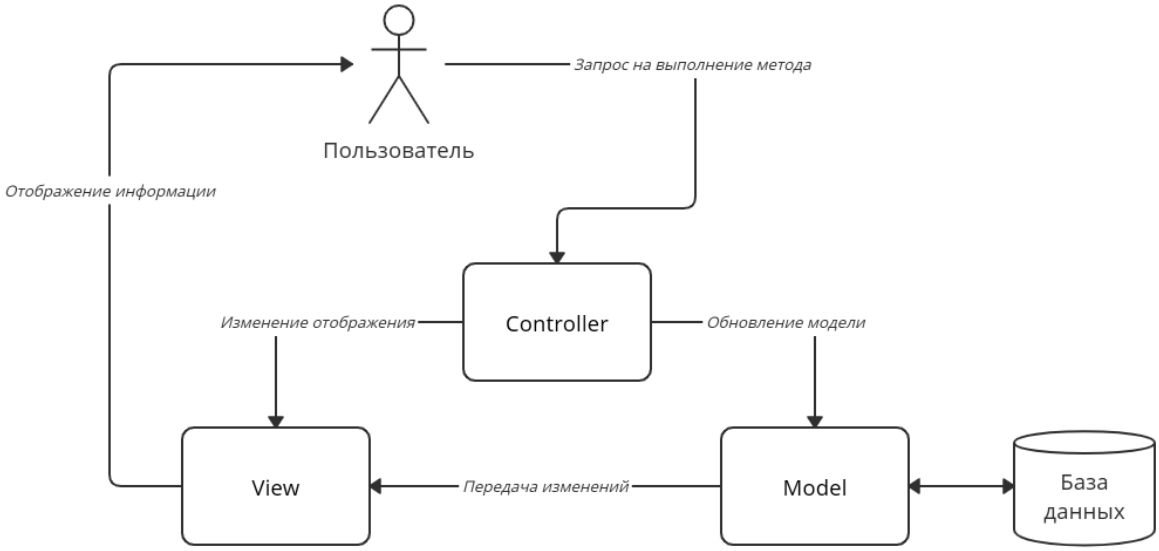


Рисунок 3.5 – Архитектура проекта

На этапе подготовки к разработке информационной системы необходимо для каждого блока программного обеспечения описать процесс взаимодействия пользователя с системой, а также её внутренних компонент между собой. На рисунках 3.6 – 3.10 приведены UML-диаграммы последовательности для различных блоков проектируемой системы.

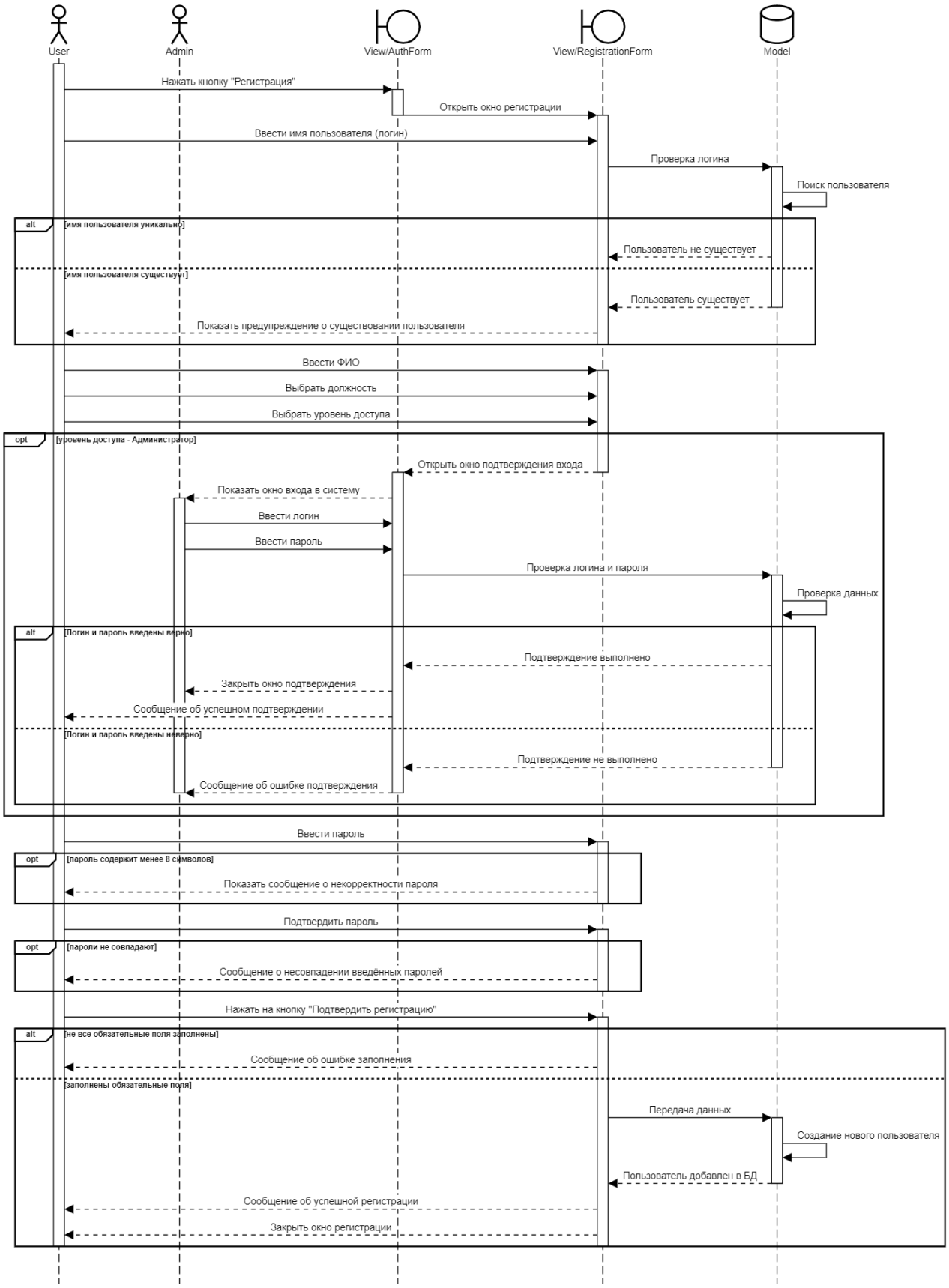


Рисунок 3.6 – Диаграмма последовательности окна регистрации

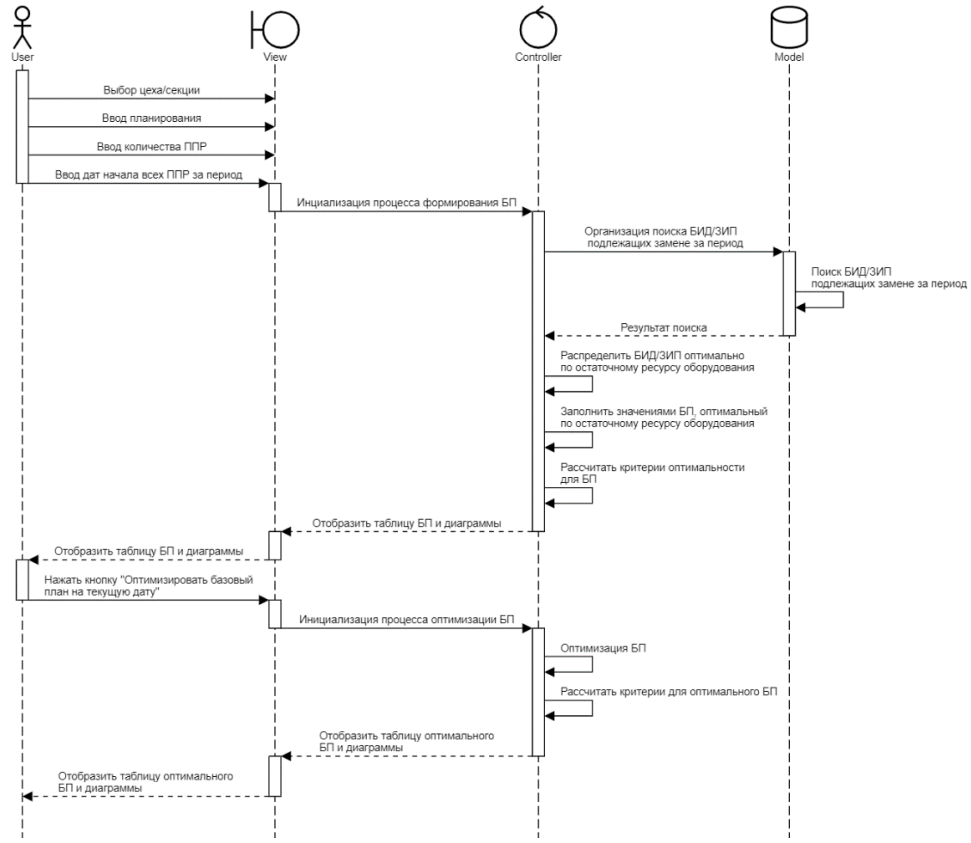


Рисунок 3.7 – Диаграмма последовательности вкладки «Экономия ресурсов от своевременной замены»

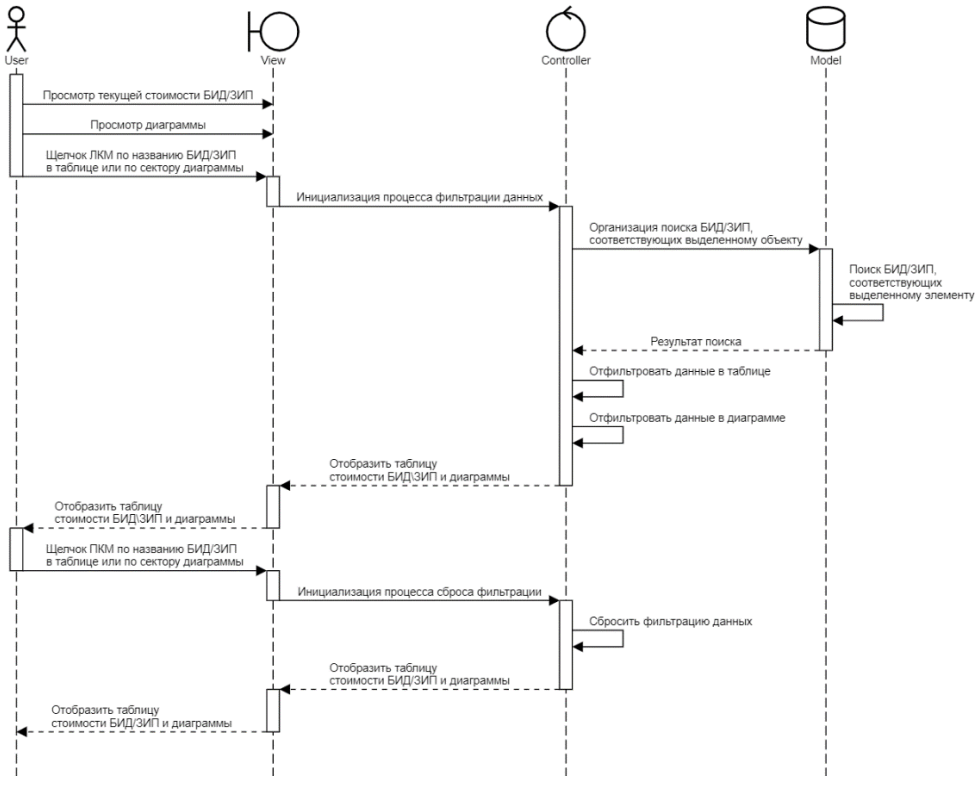


Рисунок 3.8 – Диаграмма последовательности вкладки «Текущая стоимость БИД/ЗИП»

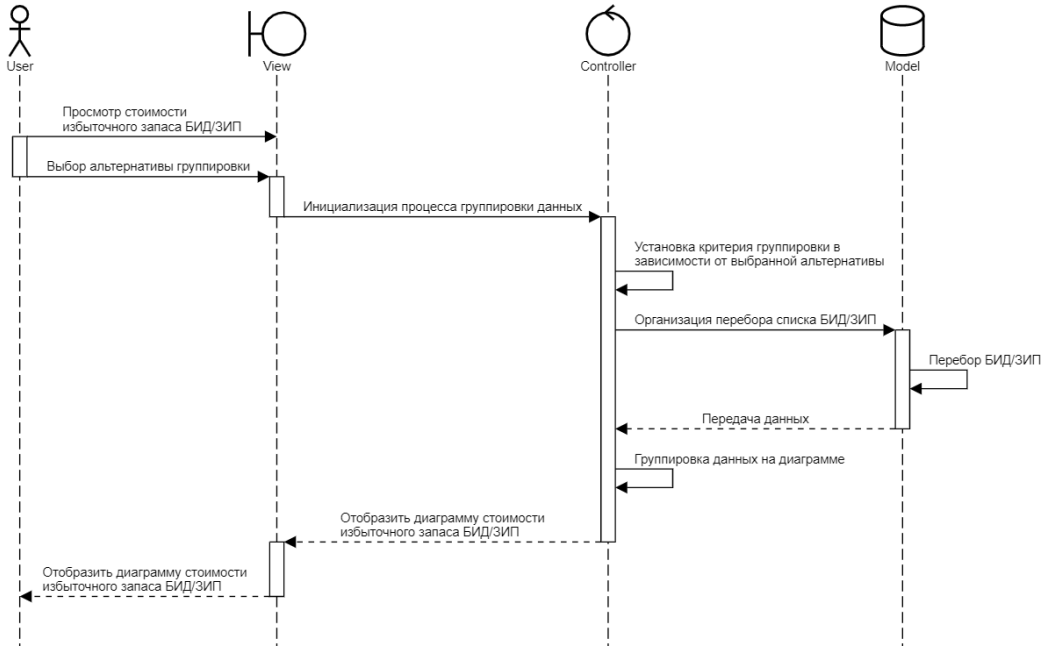


Рисунок 3.9 – Диаграмма последовательности вкладки «Стоимость избыточного запаса БИД/ЗИП»

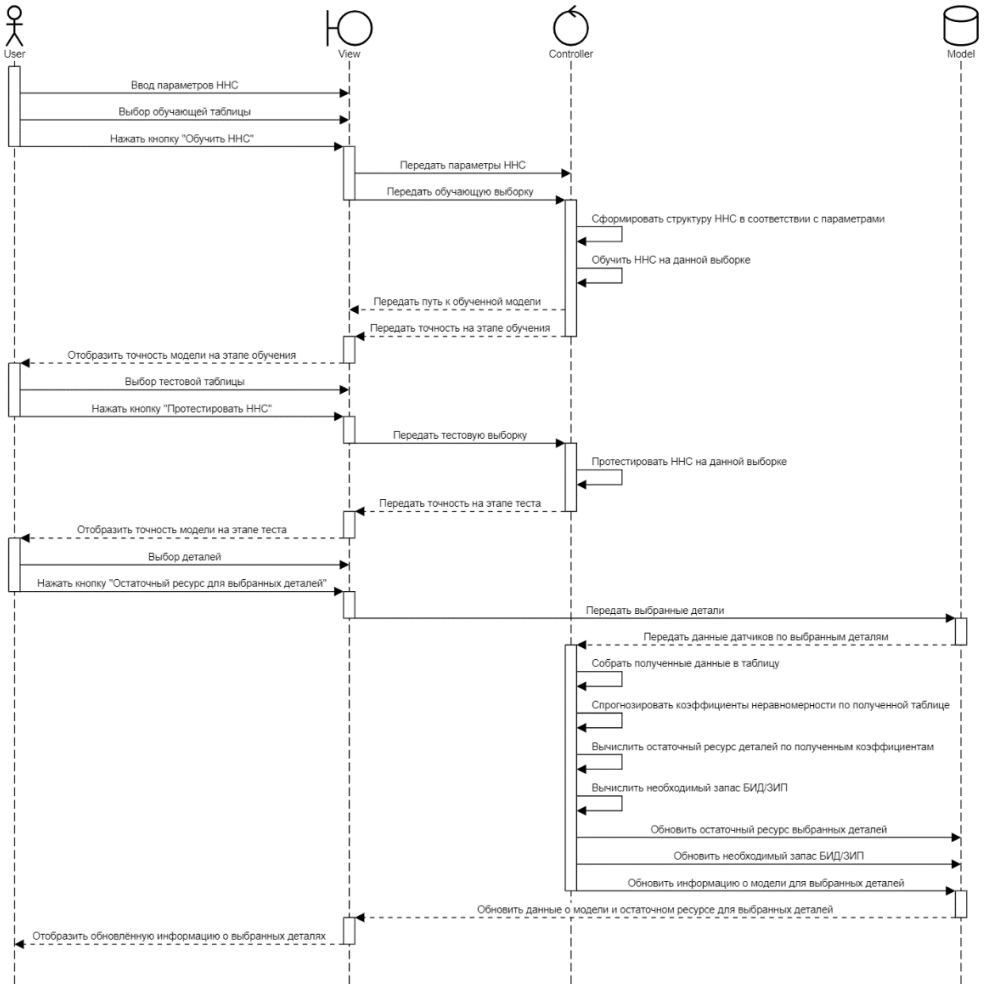


Рисунок 3.10 – Диаграмма последовательности окна «Имитационные модели и ННС»

3.4. Проектирование базы данных

3.4.1. Анализ предметной области

Для разработки базы данных программного обеспечения были выделены 4 группы сущностей: структура производства, сущности для работы нейро-нечёткой сети, сущности для составления графика ППР и сущности, относящиеся к сотрудникам на производстве.

Структура производства включает в себя 9 сущностей:

1. Производственная площадка - определенная предприятием группа объектов, объединенная по физическому, территориальному или логическому принципу. Она может включать производственные участки, технологические линии, гибкие производственные модули и производственные агрегаты. Пример: завод по производству специализированных жиров и маргаринов для пищевой промышленности «ЭФКО Пищевые Ингредиенты»;
2. Цех – подразделение производственной площадки выполняющее совокупность взаимосвязанных и взаимодействующих видов деятельности, преобразующая входы (сырье, полуфабрикаты) в выходы (полуфабрикаты, готовая продукция), через изменение их первоначальных физических и химических свойств. Представляет из себя комплекс машин, агрегатов, механизмов, узлов, а также аппаратов, колонн, установок, технологических линий, электротехнические и теплотехнические объекты, сети, технологические и обвязочные трубопроводы и другие устройства. Пример: цех производства немодифицированных жиров. Реализовывает процесс производства полуфабрикатов – гидрогенизированных жиров, переэтерифицированных жиров и дезодорированных жировых основ для производства жиров специализированного назначения;
3. Секция – совокупность технологических объектов, реализующая один определенный технологический процесс в цепочке производства

продукции. Например, секция гидрогенизации – производит химический процесс изменения жирнокислотного и триглицеридного состава исходного сырья путем полного или частичного присоединения водорода к ненасыщенным связям непредельных жирных кислот, входящих в состав триглицеридов природных масел и жиров;

4. Технологический объект – упорядоченная группа агрегатов и узлов технологической секции выполняющих определенную упорядоченную последовательность взаимосвязанных действий части общего технологического процесса, выполняющихся с момента возникновения исходных данных до получения требуемого результата. Например, теплообменник-деаэратор реализует процесс рекуперативного нагрева поступающего не гидрогенизированного масла, через охлаждение циркулирующего масла, прошедшего гидрогенизацию и состоит из комплекса агрегатов, узлов, запорно-регулирующей аппаратуры и КИПиА;
5. Функциональный элемент – полнофункциональный агрегат в составе технологического объекта. Например, насос, электродвигатель, теплообменник и т.д.;
6. Деталь функционального элемента – составные части агрегата, обеспечивающие его корректную работу. Например, подшипник, торцевое уплотнение, ротор электродвигателя и т.д.;
7. БИД/ЗИП – быстроизнашиваемые детали, запчасти и приборы. Уплотнения, прокладки, подшипники и т.д.;
8. Класс оборудования – группа агрегатов со схожими параметрами и комплектацией;
9. Датчик – устройство, установленное на объекте, которое измеряет определённые физические параметры. Например, температура масла, скорость вращения мешалки. В некоторых случаях физические

параметры фиксируются работником производства, либо описываются имитационной моделью.

Для подачи данных на вход нейро-нечёткой сети и функционирования алгоритма оценки остаточного ресурса элементов оборудования выделены следующие сущности:

1. Показатель SCADA – измерения, снятые с датчиков SCADA;
2. Ручное измерение – качественный показатель, зафиксированный работником производства;
3. Показатель имитационной модели – измерение физического параметра элемента оборудования, полученное с помощью имитационной модели;
4. Результат работы ННС – коэффициент нагрузки детали оборудования, полученный в результате работы нейро-нечёткой сети в конкретный момент времени;

Для составления плана предупредительных работ и функционирования алгоритма оптимального распределения деталей по датам замен было выделено 4 сущности:

1. План – график ремонтов, составленный с помощью алгоритма оптимизации распределения замен деталей по датам ППР в определённый момент времени;
2. Отчёт – зафиксированная поломка оборудования;
3. Вид ремонтной работы – составляющая часть ППР. Например, замена, разборка или сборка;
4. Комплекс ремонтов – группа ремонтных работ для конкретного класса объектов. Например, капитальный ремонт редуктора включает в себя замену, разборку и сборку редуктора, а также замену шестерни.

Для описания взаимодействия сотрудников производственной площадки с интеллектуальной системой были выделены следующие сущности:

1. Пользователь – сотрудник производственной площадки, имеющий доступ к использованию ПО;

2. Должность – служебная обязанность сотрудника;
3. Роль – уровень доступа пользователя к функционалу программного обеспечения;
4. Посещение – зафиксированный промежуток времени взаимодействия пользователя с системой.

Таким образом, в ходе описания предметной области был выделен 21 объект, необходимые для хранения в базе данных.

Выполнив анализ предметной области, выделим ключи сущностей, используемые для идентификации экземпляров сущностей:

1. Производственная площадка (Ключ – Название площадки)
2. Цех (Ключ – Название цеха)
3. Секция (Ключ – Название секции)
4. Технологический объект (Ключ – Код объекта)
5. Функциональный элемент (Ключ – Код тех. объекта, название функционального элемента)
6. Деталь функционального элемента (Ключ – Код тех. объекта, название функционального элемента, название детали)
7. БИД/ЗИП (Ключ – Код тех. объекта, название функционального элемента, название детали, название БИД/ЗИП)
8. Класс оборудования (Ключ – Название класса)
9. Измерение (Ключ – Название измерения, код тех. объекта, название функционального элемента, название детали)
10. Показатель SCADA (Ключ – Название измерения, код тех. объекта, название функционального элемента, название детали, дата и время снятия показания)
11. Ручное измерение (Ключ – Название измерения, код тех. объекта, название функционального элемента, название детали, дата и время снятия показания)

12. Показатель имитационной модели (Ключ – Название измерения, код тех. объекта, название функционального элемента, название детали, дата и время снятия показания)
13. Результат работы ННС (Ключ – Код тех. объекта, название функционального элемента, название детали, дата и время прогнозирования)
14. План (Ключ – Дата формирования плана, название комплекса ремонтных работ, код тех. объекта, название функционального элемента, название детали, название БИД/ЗИП)
15. Отчёт (Ключ – Код тех. объекта, название функционального элемента, название детали, дата и время поломки)
16. Вид ремонтной работы (Ключ – Название вида ремонтной работы)
17. Комплекс ремонтов (Ключ – Название комплекса ремонтных работ, код тех. объекта, название функционального элемента, название детали, название БИД/ЗИП)
18. Пользователь (Ключ – Логин пользователя)
19. Должность (Ключ – Название должности)
20. Роль (Ключ – Название уровня доступа)
21. Посещение (Ключ – Логин пользователя, дата и время входа)

Определим связи между указанными сущностями:

1. Площадка ИМЕТЬ Цех
2. Цех РАСПОЛАГАТЬ Секция
3. Секция СОДЕРЖАТЬ Технологический объект
4. Технологический объект ВКЛЮЧАТЬ Функциональный элемент
5. Функциональный элемент СОСТОЯТЬ Деталь функционального элемента
6. Деталь функционального элемента ОТНОСИТЬ БИД/ЗИП
7. Деталь функционального элемента СНИМАТЬ Измерение
8. Измерение ФИКСИРОВАТЬ Показатель SCADA
9. Измерение ЗАПИСЫВАТЬ Ручное измерение

10.Имитационная модель ВЫЧИСЛЯТЬ Измерение

11.Класс РАЗДЕЛЯТЬ Деталь

12.Результат работы ННС ОЦЕНИВАТЬ Деталь

13.Деталь ПОЛУЧАТЬ Отчёт

14.Вид ремонтной работы ДЕЛАТЬ БИД/ЗИП

15.Комплекс ремонтных работ ВЫПОЛНЯТЬ БИД/ЗИП

16.Комплекс ремонтных работ ВМЕЩАТЬ Вид ремонтной работы

17.План ЗАКЛЮЧАТЬ Комплекс ремонтных работ

18.Пользователь РАБОТАТЬ Производственная площадка

19.Пользователь ЗАНИМАТЬ Должность

20.Пользователь ОБЛАДАТЬ Роль

21.Пользователь ОСТАВЛЯТЬ Посещение

3.4.2. Построение инфологической модели

Степень связи ИМЕТЬ – 1:M (1 ко многим) [17], поскольку одна площадка может иметь несколько цехов, однако один цех располагается только на одной площадке. Класс принадлежности связи: O-O, т.к. площадка обязательно имеет на своей территории цех, как и цех обязательно должен быть расположен на площадке.

Степень связи РАСПОЛАГАТЬ – 1:M, потому что на территории цеха может быть несколько секций, но секция имеет только одну площадку. Класс принадлежности связи: O-O, поскольку в цехе есть как минимум одна секция, а секция непременно находится на территории цеха.

Степень связи СОДЕРЖАТЬ – 1:M, ведь на секции есть несколько технологических объектов, но технологический объект есть только на одной секции. Класс принадлежности связи: O-O, потому что секция обязана включать технологический объект, а технологический объект должен располагаться на какой-либо секции.

Степень связи ВКЛЮЧАТЬ – 1:M, потому что технологического объекта может быть несколько функциональных элементов, однако функциональный элемент принадлежит одному конкретному технологическому объекту. Класс

принадлежности связи: О-О, т.к. технологический объект имеет состоит из нескольких функциональных элементов, как и функциональные элементы обязательно принадлежат технологическому объекту.

Степень связи СОСТОЯТЬ – 1:М, поскольку у функционального элемента могут быть детали функционального элемента, но все детали функционального элемента имеют отношение только к одному функциональному элементу. Класс принадлежности связи: О-О. Хоть функциональный элемент и может не иметь составные части, но для упрощения формирования базы данных допустим, что составная часть такого функционального элемента есть сам функциональный элемент. Разумеется, составная часть функционального элемента принадлежит функциональному элементу.

Степень связи ОТНОСИТЬ – 1:М, т.к. деталь функционального элемента не исключает содержание нескольких составных элементов, т.е. БИД/ЗИП, однако БИД/ЗИП относятся исключительно к одной детали функционального элемента. Класс принадлежности связи: О-О. Деталь функционального элемента может не иметь составных частей и быть БИП/ЗИП самим по себе, но для упрощения формирования базы данных допустим, что деталь функционального элемента сама является БИП/ЗИП, если она быстро изнашивается. При этом БИП/ЗИП обязательно – составной элемент детали функционального элемента.

Степень связи СНИМАТЬ – М:М, т.к. с одной детали функционального элемента может быть снято несколько измерений, а одно измерение может характеризовать сразу несколько деталей. Класс принадлежности: Н-О. Если измерение существует, то он непременно относится к какой-либо детали, однако с детали может и не сниматься какое-либо измерение.

Степень связи ФИКСИРОВАТЬ – 1:М, т.к. одно измерение может быть сделано на основании нескольких датчиков SCADA, однако один конкретный датчик SCADA характеризует только одно измерение. Класс принадлежности: Н-О, т.к. измерение может не быть описано с помощью датчика SCADA, однако показатель датчика SCADA обязательно относится к какому-то конкретному измерению.

Степень связи ЗАПИСЫВАТЬ – 1:M, т.к. одно измерение может быть сделано на основании нескольких ручных измерений, однако один конкретный оперативный отчёт характеризует только одно измерение. Класс принадлежности: H-O, т.к. измерение может не быть описано с помощью оперативных отчётов, однако оперативный отчёт обязательно относится к какому-то конкретному измерению.

Степень связи ВЫЧИСЛЯТЬ – M:1, т.к. одно измерение может быть сделано на основании нескольких расчётов имитационных моделей, однако один конкретный расчёт имитационной модели относится лишь к одному измерению. Класс принадлежности: O-H, т.к. измерение может не быть описано с помощью имитационной модели, однако результат расчёта имитационной модели обязательно относится к какому-то конкретному измерению.

Степень связи РАЗДЕЛЯТЬ – 1:M, т.к. один класс может включать несколько деталей, а одна конкретная деталь включена лишь в один конкретный класс деталей. Класс принадлежности: O-O, т.к. если класс существует, то в нём должны быть детали, как и деталь непременно должна быть включена в какой-либо класс.

Степень связи ОЦЕНИВАТЬ – M:1, т.к. для одной детали может быть проведена оценка нагрузки несколько раз, но одна конкретная оценка нагрузки рассчитывается лишь для одной детали. Класс принадлежности: O-H, т.к. для детали может не проводиться оценка коэффициента нагрузки, однако если оценка была проведена, то точно для какой-либо детали.

Степень связи ПОЛУЧАТЬ – M:M, т.к. один отчёт о поломке может содержать информацию о нескольких деталях, как и одна деталь в архиве может содержать серию случаев поломки. Класс принадлежности: H-O, т.к. деталь могла не ломаться, однако если поломка была зафиксирована, то непременно затронула какую-либо деталь.

Степень связи ДЕЛАТЬ – M:M, т.к. один вид ремонтных работ может относиться к разным БИД/ЗИП, как и на одном БИД/ЗИП может проводиться несколько видов ремонтных работ. Класс принадлежности: O-O, т.к. любой

БИД/ЗИП периодически надо заменять, как и вид ремонта относится хотя бы к одному БИД/ЗИП.

Степень связи ВЫПОЛНЯТЬ – М:1, т.к. один комплекс ремонтных работ относится лишь к одному БИД/ЗИП, однако для одного БИД/ЗИП может быть предусмотрено несколько комплексов ремонтных работ. Класс принадлежности: О-О, т.к. для любого БИД/ЗИП периодически надо проводить полный комплекс работ, как и комплекс работ относится хотя бы к одному БИД/ЗИП.

Степень связи ВМЕЩАТЬ – М:М, т.к. один конкретный ремонт может входить в несколько комплексов, как и комплекс ремонтных работ как правило состоить из нескольких видов ремонтов. Класс принадлежности: О-О, т.к. любой вид ремонтной работы включён в какой-либо комплекс, как и комплекс ремонтных работ включает хотя бы один вид ремонтных работ.

Степень связи ЗАКЛЮЧАТЬ – М:М, т.к. один комплекс ремонтных работ может быть включён в несколько планов ППР, как и один план ППР обычно содержит несколько комплексов ремонтных работ. Класс принадлежности Н-О, т.к. комплекс может существовать, но не быть включён в какой-либо план ППР, однако если план ППР существует, то в него входит хотя бы один комплекс ремонтных работ.

Степень связи РАБОТАТЬ – М:1, т.к. один сотрудник занимает должность только на одной производственной площадке, однако на производственной площадке всегда несколько сотрудников. Класс принадлежности: О-О, т.к. как минимум один сотрудник производственной площадке должен иметь доступ к системе, как и сотрудник обязательно закреплён за какой-либо производственной площадкой.

Степень связи ЗАНИМАТЬ – М:1, т.к. одну должность могут занимать несколько сотрудников, но один конкретный сотрудник назначен лишь на одну должность. Класс принадлежности: О-Н, т.к. если пользователь есть в системе, то он обязательно занимает должность на производстве, однако для существующей должности в системе может не быть ни одного зарегистрированного пользователя.

Степень связи ОБЛАДАТЬ – M:M, т.к. один пользователь может иметь несколько уровней доступа к системе, как и один уровень доступа может быть предоставлен нескольким пользователям. Класс принадлежности: O-H, т.к. у пользователя обязательно должен быть какой-либо уровень доступа, однако для существующего уровня доступа может не быть зарегистрированного сотрудника.

Степень связи ОСТАВЛЯТЬ – 1:M, т.к. один пользователь как правило заходит в систему несколько раз, однако одно конкретное посещение принадлежит только одному пользователю. Класс принадлежности: H-O, т.к. сотрудник может быть зарегистрирован, но ни разу не вошёл в систему, однако если вход в систему был совершён, то непременно каким-либо пользователем.

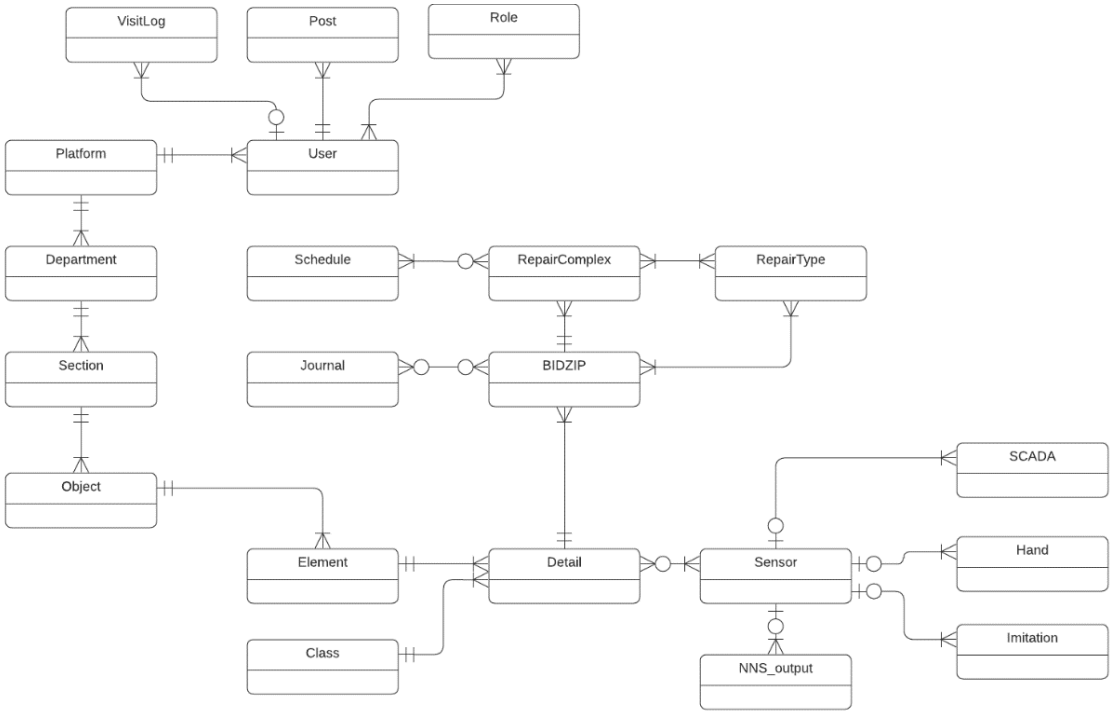


Рисунок 3.11 – ER-диаграмма инфологической модели

3.4.3. Физическая модель базы данных

На этапе разработки физической модели базы данных в таблицах вводятся идентификаторы в качестве первичных ключей.

Полученные таблицы с указанием названий атрибутов, их типов данных, допустимых значений, значений по умолчанию и ключей приведены в Приложении Д.

3.4.4. Анализ сложности физической схемы базы данных

Оценку сложности базы данных можно проводить на различных стадиях разработки. Однако наиболее точная оценка может быть получена по её физической схеме.

Существует несколько подходов измерения сложности физической схемы базы данных. Рассмотрим несколько из них.

Пусть W_i – вес сложности i-й таблицы базы данных. Данная величина вычисляется по формуле:

$$W_i = \alpha_1 A_i + \alpha_2 K_i + \alpha_3 I_i + \alpha_4 F_i + \alpha_5 T_i \quad (3.1)$$

где A_i – количество атрибутов в i-й таблице; K_i – количество ключей в i-й таблице; I_i – количество неуникальных индексов в i-й таблице; F_i – количество внешних ключей в i-й таблице, T_i – количество различных типов данных в i-й таблице, α_j – коэффициент, учитывающий степень влияния j-й метрики на сложность физической схемы базы данных.

Коэффициенты α_j были определены в [18] с помощью процедуры Саати.

Таблица 3.1 – Коэффициенты степени влияния метрик на сложность БД

Обозначение	**Метрика**	***α***
A	Количество атрибутов	0.105
K	Количество ключей	0.315
I	Количество неуникальных индексов	0.030
F	Количество внешних ключей	0.489
T	Количество различных типов данных	0.061

Сложность физической схемы базы данных равна сумме весов сложности всех таблиц базы данных:

$$C = \sum_{i=1}^{n} W_i \tag{3.2}$$

Таблица 3.2 – Метрические характеристики физических схем таблиц БД

i	**Таблица**	A_i	K_i	I_i	F_i	T_i	W_i
1	Platform	4	1	0	0	2	0,857
2	Department	8	1	0	1	3	1,827
3	Section	5	1	0	1	2	1,451
4	Object	6	1	1	1	2	1,586
5	Class	4	1	0	0	2	0,857
6	Element	5	1	1	1	2	1,481
7	Detail	6	1	1	2	2	2,075
8	BIDZIP	8	1	1	1	4	1,918
9	Sensor	6	1	0	0	3	1,128
10	Sensor_Detail	4	2	0	2	2	2,15
11	SCADA	3	2	1	1	3	1,647
12	Hand	3	2	1	1	3	1,647
13	Imitation	3	2	1	1	3	1,647
14	NNS_output	3	2	0	1	3	1,617
15	Journal_table	3	1	1	0	3	0,843
16	Journal_table_Detail	3	2	0	2	2	2,045
17	Repair_type	2	1	0	0	2	0,647
18	Repair_type_BIDZIP	5	1	0	2	2	1,94
19	Repair_complex	3	1	0	1	2	1,241
20	Repair_complex_to_Type	2	2	0	2	1	1,879
21	Schedule	2	1	1	0	2	0,677
22	Schedule_Repair_complex	5	2	0	2	3	2,316
23	Post	2	1	0	0	2	0,647
24	User_table	8	1	0	2	2	2,255
25	Role	2	1	0	0	2	0,647
26	User_Role	2	2	0	2	1	1,879
27	VisitLog	4	2	0	1	2	1,661
Сумма		111	37	9	27	62	40,565

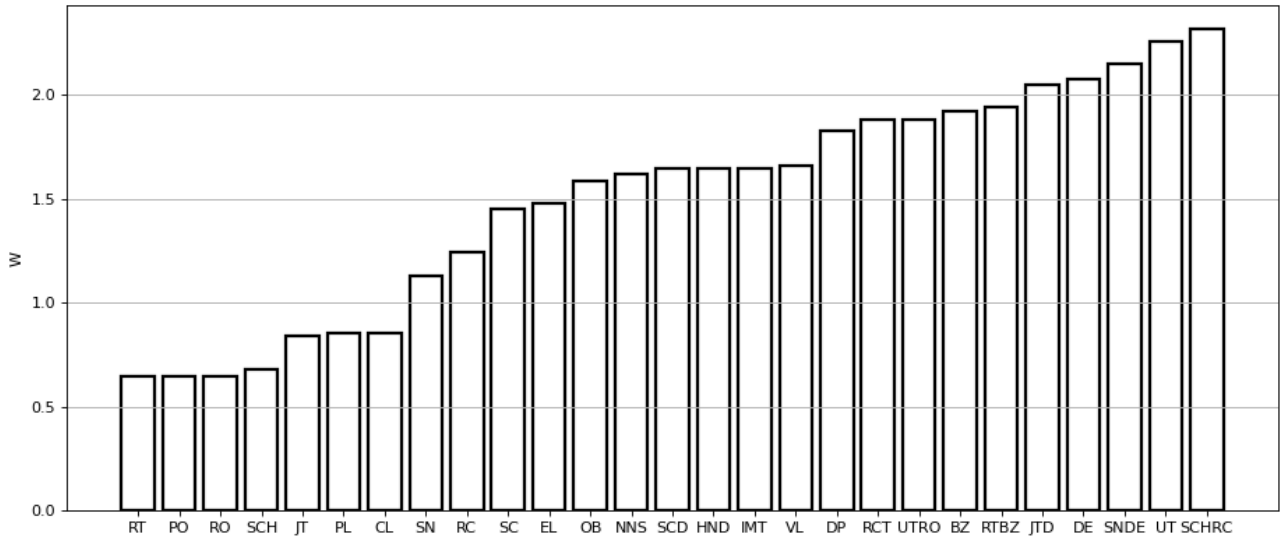


Рисунок 3.12 – Значения сложности физической схемы БД

Сложность физической схемы базы данных составила 40,565. Оценённый показатель распределён неравномерно по таблицам БД (рис. 2.2). Это говорит о том, что сложность схемы обусловлен не только количеством проектов в БД, но и влиянием отдельных таблиц. Данный показатель может быть использован при добавлении новых баз данных в интеллектуальную систему «Умные ремонты», поскольку характеризует трудоёмкость работ разработчика над проектом.

3.5. Графический интерфейс

Графический интерфейс пользователя (GUI) был создан с помощью набора расширений фреймворка Qt для языка программирования Python. При разработке GUI были соблюдены следующие принципы проектирования: масштабируемость и интуитивность. [19]

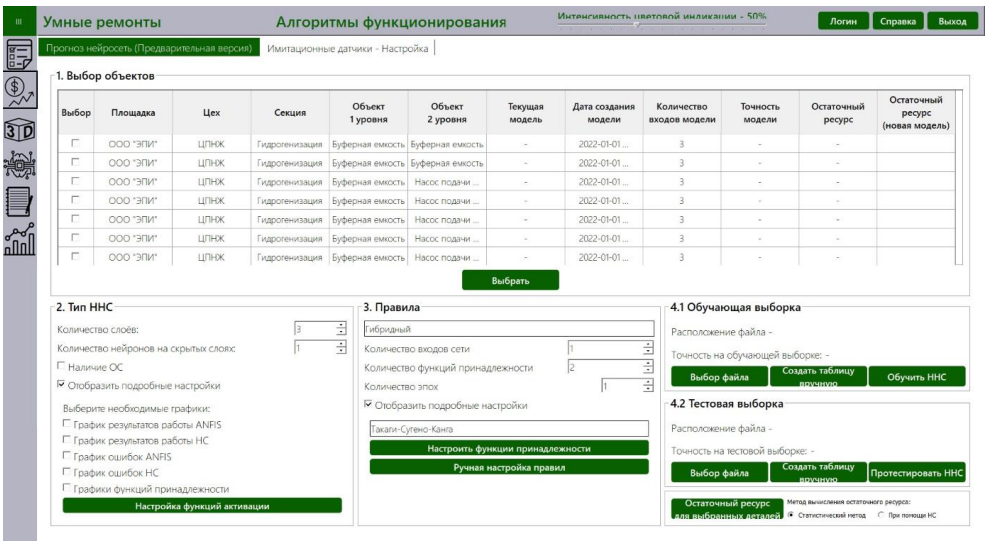


Рисунок 3.13 – Интерфейс блока «Имитационные модели и ННС»

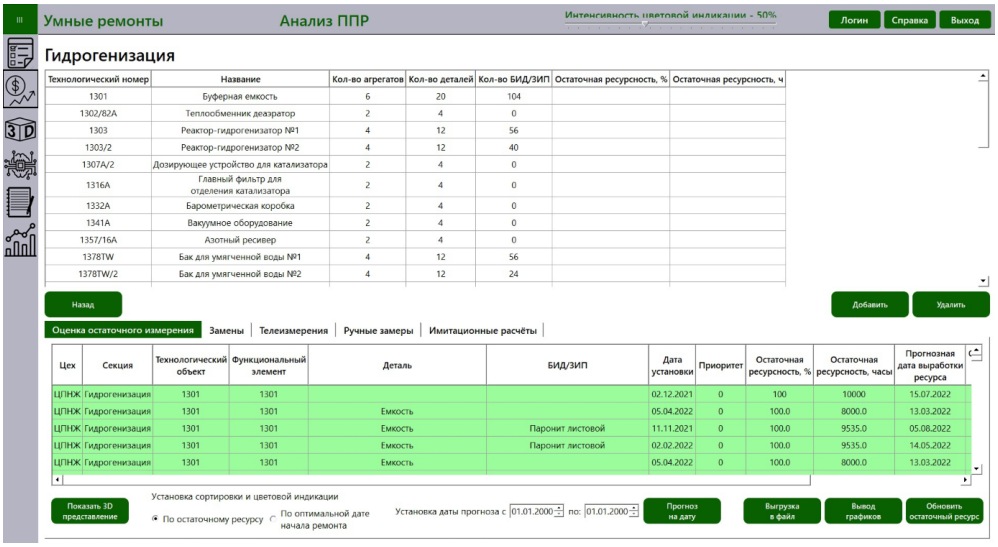


Рисунок 3.14 – Интерфейс блока «База данных»

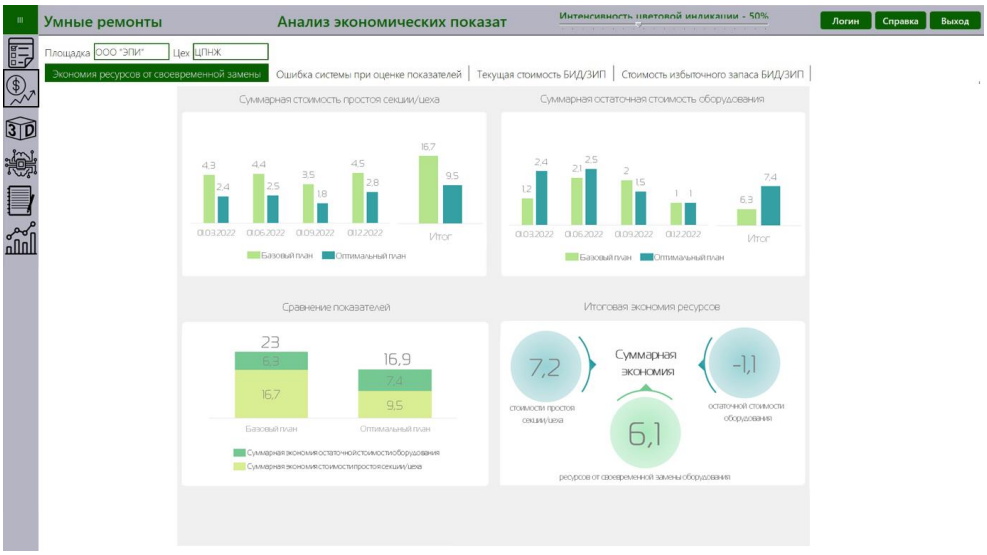


Рисунок 3.15 – Интерфейс блока «Анализ экономических показателей»

4. Концепция стартап-проекта

4.1. Описание продукта как результата НИР

В рамках данной работы разработано программное обеспечение, которое:

1. Прогнозирует дату полного износа деталей производственного оборудования;
2. Формирует рекомендацию, в какую дату экономически эффективно выполнять ремонтные работы отдельных деталей. Критерии эффективности: минимальное время простоя оборудования и минимальная остаточная стоимость заменяемых деталей;
3. Составляет график планово-предупредительных ремонтов согласно полученным данным.

4.2. Интеллектуальная собственность

Объектом интеллектуальной собственности является программа для ЭВМ. Отношения, возникающие в связи с правовой охраной и использованием программ для ЭВМ, регулирует Гражданский кодекс РФ, часть 4, ст. 1261 и ст. 1262.

Полностью запатентовать программное обеспечение невозможно, т.к. ПО не является изобретением согласно ст. 1350 п. 5 ГК РФ. Разрабатываемая программа состоит из алгоритма, исходного кода, базы данных и интерфейса. Только алгоритм и интерфейс ПО признаются патентоспособными. База данных и исходный код охраняются авторским правом. Ключевое различие заключается в том, что авторское право защищает автора от копирования программы, а патентное право защищает идею программы в целом [20].

Таким образом, для защиты интеллектуальной собственности принято решение о регистрации исходного кода (программы для ЭВМ), базы данных и патентование алгоритма и интерфейса программного обеспечения.

4.3. Целевые сегменты потребителей

Наша целевая аудитория – это крупные предприятия молочной и масложировой промышленности с автоматизированным производством, которые расположены в Сибирском федеральном округе. Также предприятия, являющиеся нашей целевой аудиторией, должны иметь установленную систему датчиков SCADA или иметь возможность проведения оперативных измерений при помощи измерительных приборов используемых работниками цехов. Такой уровень автоматизации диспетчерского управления позволит обеспечить передачу информации, полученной с датчиков, в разрабатываемое ПО.

В структуре предприятий, являющихся нашим целевым сегментом можно выделить несколько заинтересованных лиц. Во-первых, это представители отделов контроля качества оборудования, которые заинтересованы в том, чтобы оборудование как можно реже выходило из строя, и предлагаемое решение позволяет предотвратить возникновение таких случаев. Также в нашем решении заинтересованы представители отдела закупок, потому что разрабатываемое ПО формирует график ремонтов таким образом, чтобы объём закупок был минимальным, это позволит снизить затраты, связанные с закупкой запчастей, а также уменьшить количество запасных элементов, хранящихся на складах. Таким образом, в следствии применения ПО снизятся эксплуатационные затраты предприятия и увеличится его общая прибыль, в чём заинтересованы директора компаний.

4.4. Объём и ёмкость рынка

Проведём оценку рынка двумя методами: снизу-вверх и сверху-вниз. Для первого метода возьмём данные из Справочника организаций и предприятий России [21] о количестве предприятий производящих молочную продукцию и находящихся в СФО – таких предприятий 164. Исходя из представленных описаний данных предприятий около половины из них автоматизированы достаточно, для применения предлагаемого решения – это 82 предприятия.

Таким образом TAM составляет $164 \cdot 0,5 \cdot 475000 = 38950000$ руб., исходя из средней стоимости разрабатываемого продукта (475 000 руб.).

Далее, оценим сколько из предприятий, попавших в TAM, имеют достаточно большую выручку и дорогостоящее оборудование, чтобы наше решение оказалось выгодным для них. Исходя из описаний, таких предприятий около 20% (примерно 17 производств), поэтому SAM составляет $164 \cdot 0,5 \cdot 0,2 \cdot 475000 = 7790000$ руб.

Исходя из географии расположения предприятий, примерно с четвертью из них мы сможем установить сотрудничество в течении года, таким образом SOM составляет $164 \cdot 0,5 \cdot 0,2 \cdot 0,25 \cdot 475000 = 1947500$ руб.

Проведём оценку методом сверху-вниз. В соответствии с исследованием [22], прогнозируемый объём всех Process Mining решений в России на 2022 год составляет 80 043 671 руб., при этом приблизительная доля молочной промышленности по данным наших конкурентов составляет 5%, таким образом TAM равен $80043671 \cdot 0,05 = 4002183,55$ руб.

Примерно треть предприятий из TAM географически находятся в СФО, соответственно SAM составляет $4002183,55 \cdot 0,33 = 1320720,57$ руб. При этом, два наших конкурента работали с региональными предприятиями, значит могут прийти и на рынок СФО в том числе, таким образом SOM составляет $1320720,57 / (2+1) = 440240,19$ руб.

По результатам оценки рынка, наиболее оптимистичный прогноз получен методом снизу-вверх и объём рынка на год составляет 1 947 500 руб.

Таблица 4.1 – Сравнение двух методов оценки рынка

Снизу-вверх	Сверху вниз
TAM	
Всего предприятий молочной продукции в СФО - 164. Около 50% из них автоматизированны. Средняя цена нашего ПО 475 000. **164*0,5*475 000 = 38 950 000**	Прогнозируемый объём рынка всех Process Mining решений в России на 2022 - 80 043 671. Приблизительная доля молочной промышленности - 5%. **80 043 671*0,05 = 4 002 183,55**
SAM	
Около 20% из автоматизированных предприятий достаточно крупные, чтобы наше решение было им выгодно. **164*0,5*0,2*475 000 = 7 790 000**	Приблизительная доля предприятий молочной промышленности из СФО, готовых приобретать Process Mining решения - 33%. **4 002 183,55*0,33 = 1 320 720,57**
SOM	
Приблизительная доля предприятий, с которыми мы сможем установить сотрудничество за год, исходя из географии - 25%. **164*0,5*0,2*0,25 = 1 947 500**	Конкуренты, которые потенциально могут сделать аналогичное ПО для данных предприятий - 2 компании + мы, итого 3 компании на одном рынке в одной области. **1 320 720,57 / 3 = 440 240,19**

4.5. Анализ современного состояния и перспектив развития отрасли

За последние годы в промышленности наблюдается активное внедрение технологий Индустрии 4.0 (технологии искусственного интеллекта, Интернет вещей, автоматизация бизнес-процессов). На данный момент крупные компании (например, государственная корпорация «Росатом») самостоятельно разрабатывают интеллектуальные системы по оптимизации технического обслуживания и ремонтов на базе собственных ИТ-интеграторов. На российском рынке существует несколько компаний, которые индивидуально для каждого заказчика адаптируют программное обеспечение, позволяющее решить проблему оптимального планирования ремонтов и оценки технического состояния оборудования. Однако данная процедура распространена прежде всего на западе страны, поскольку требует непосредственного присутствия

разработчиков на предприятии-заказчике. По этой причине невозможно охватить не только все сферы применения решения, но и предприятия, расположенные на большом расстоянии от компаний, предоставляющих услуги по разработке.

Сферы применения интеллектуальной системы по оптимизации включают нефтегазовую, химическую, горно-металлургическую, энергетическую, фармацевтическую и пищевую промышленности. Предприятия перечисленных сегментов располагаются на территории всей России. Таким образом, направление автоматизации бизнес-процессов в области управления техническим обслуживанием и ремонтами имеет перспективы развития.

4.6. Планируемая стоимость продукта

Для определения планируемой стоимости годовой лицензии программного обеспечения была рассчитана себестоимость по затратному методу ценообразования.

Разработка программного обеспечения требует привлечения следующих специалистов: специалист в области машинного обучения, backend-разработчик и frontend-разработчик. Стоимость услуг специалиста в области машинного обучения оценивается в 50 000 руб/месяц, backend и frontend-разработчиков уровня Junior – в 60 000 руб/месяц.

Стоимость сервера для хранения данных составляет 110 182 рубля. Срок службы сервера составляет 5 лет и относится ко второй амортизационной группе. Срок полезного использования сервера устанавливается в пределах от 2 лет и одного месяца до 3 лет. Примем среднее значение срока полезного использования – 2,5 года. Таким образом, ежегодные амортизационные отчисления составляют 44 072,8 рублей. Услуги по установке и настройке сервера для хранения данных составляют 7500 рублей.

Разработка программного обеспечения предполагает дистанционную занятость и не предусматривает аренду офисного помещения, а также подразумевает наличие персонального компьютера у разработчиков.

Длительность создания программного продукта для одного заказчика оценивается в 2 месяца. При последовательном выполнении заказов и текущем кадровом составе предполагается совершить 6 продаж в первый год. Данный срок обусловлен наличием базового ПО, разработанного авторами в результате выполнения ВКР, что позволяет довольно быстро адаптировать сервис под новую компанию. Расчёт себестоимости представлен в таблице X.

Таблица 4.2 – Себестоимость товара

Оборудование, тыс. руб	264,437
Заработная плата и услуги сторонних организаций, тыс. руб	2 040,45
Итого на 6 ПО, тыс. руб	2 304,887
Себестоимость единицы продукции, тыс. руб	384,147

Примем прибыль от продажи годовой лицензии равной 30% в первый год пользования, тогда планируемая стоимость годовой лицензии составляет 499 391,1 рублей. Таким образом, прибыль за продажу 6 лицензий без учёта налога составит 691 464,6 рублей.

4.7. Конкурентные преимущества создаваемого продукта

Для проведения конкурентного анализа были выбраны наиболее популярные коммерческие решения по интеллектуальному мониторингу состояния производственного оборудования и оптимизации технического обслуживания и ремонтов. Сведения о продуктах SIMPLE-SCADA, решении от Digital Design, «Process Mining для ТОиР» от RAMAX Group и решении от Factory5 были взяты из открытых источников данных компаний и презентаций их проектов.

В качестве наиболее важных критериев для сравнения были выделены следующие факторы: масштабируемость, индивидуализация, учёт времени

постоя и остаточной стоимости оборудования при формировании оптимального графика ППР, а также вид принятия решений (ручное или автоматизированное). Масштабируемость важна для заказчика, поскольку позволяет ему самостоятельно вносить изменения в структуру предприятия без привлечения разработчиков. Индивидуализация же важна на этапе разработки, т.к. позволяет сократить время на адаптацию продукта под новый заказ. Учёт времени простоя и остаточной стоимости оборудования является наиболее значимым фактором, поскольку оказывает сильное влияние на оптимизацию бизнес-процесса. Вид принятия решений важен, т.к. автоматизированное решение сокращает временные и трудовые ресурсы на формирование плана ремонтов, а ручной формат позволяет принимать гибкие решения в случае непредвиденных обстоятельств (например, внезапная авария, экстренный ремонт и т.п.).

Таблица 4.3 – Конкурентный анализ

Решение / Критерий	Разрабатываемое решение	SIMPLE-SCADA	Digital Design	Process Mining для ТОиР (RAMAX)	Factory5
Масштабируемость	+	+	–	–	–
Индивидуализация	+	+	+	+	+
Сокращение времени простоя оборудования	+	–	+	+	–
Сокращение остаточной стоимости оборудования	+	–	–	+	–
Принятие решений	ручное и автоматизированное	ручное	автоматизированное	ручное и автоматизированное	ручное и автоматизированное

Таким образом, главным конкурентом согласно выделенным критериям является решение от RAMAX Group, а главным преимуществом по сравнению с данным продуктом оказалась масштабируемость системы.

4.8. Бизнес-модель проекта. Производственный план и план продаж

Приведём схематичное описание бизнес-модели проекта. Для этого воспользуемся моделью А. Остервальда, представляющую собой схему из 9 блоков, описывающих различные бизнес-процессы проекта.

Таблица 4.4 – Бизнес-модель по А. Остервальду

<table>
<tr>
<td rowspan="2">Ключевые партнёры
Молодые IT-компании, сотрудников которых можно привлечь к разработке продуктов.</td>
<td>Ключевые виды деятельности
Разработка программного обеспечения;
Техническая поддержка.</td>
<td rowspan="2">Ценностные предложения
Прогнозирование даты выхода оборудования из строя;
Расчёт экономически оптимальной даты ремонта оборудования.</td>
<td>Взаимоотношения с клиентами
Совместное создание;
Персональная техническая поддержка.</td>
<td rowspan="2">Потребительские сегменты
Отделы контроля качества оборудования, отделы закупок, директора компаний масложировой и молочной промышленности.</td>
</tr>
<tr>
<td>Ключевые ресурсы
Финансы (з/п сотрудникам);
Средства для размещения БД на сервере</td>
<td>Каналы сбыта
Основной канал сбыта: информационный – участие в форумах, конференциях.</td>
</tr>
<tr>
<td colspan="3">Структура издержек
Постоянные: оплата за размещение базы данных на сервере, заработная плата сотрудников тех. поддержки на начальном этапе реализации ПО.
Переменные: заработная плата сотрудников, когда заказов окажется достаточно много.</td>
<td colspan="2">Потоки поступления доходов
Продажа годовой лицензии программного обеспечения, продление лицензии.</td>
</tr>
</table>

4.9. Стратегия продвижения продукта на рынок

Целью продвижения проекта на рынок является привлечь не менее 6 горячих лидов в первый год работы за счёт посещения мероприятий (конференции, выставки и т.д.), которыми интересуются представители предприятий, входящих в целевую аудиторию, а также посредством публикаций в тематических журналах. Приоритетными точками касания в данном случае являются личные встречи, переписки, звонки с ЛПР.

Определим то, как должен выглядеть путь клиента:

1. Увидел статью в журнале/выступление на конференции;
2. Сохранил наши контакты;
3. Позвонил/написал/встретился для обсуждения особенностей решения;
4. Купил продукт.

Первый и третий пункты цепочки действий клиента варьируется в зависимости от того, каким образом потенциальный клиент узнал о нашем продукте, что определяется тем или иным способом продвижения. Также стоит отметить, что после покупки продукта взаимодействие с клиентом может продолжиться, если он продлит лицензию на пользование ПО внесением оплаты через год или купит новый продукт на другую производственную площадку своей компании.

5. Социальная ответственность

5.1. Введение

Объектом разработки данной ВКР является программное обеспечение для формирования графика планово-предупредительных ремонтных работ. Методы глубинного обучения, использованные при разработке интеллектуальной системы, дают возможность оценить реальную загрузку оборудования на производстве и определить срок выработки ресурса деталей. Алгоритм оптимизации распределения деталей по датам ППР позволяет совершать замены с наименьшим простоем и остаточной стоимостью единицы оборудования.

Проект выполняется на персональном компьютере (ПК), поэтому в данном разделе проводится анализ опасных и вредных факторов при работе с ПК, влияния этих факторов на окружающую среду и мероприятий по её защите.

Предметом исследования является рабочая зона разработчика, включая компьютерный стол, ПК, клавиатуру, компьютерную мышь и стул. Работы выполнялись в компьютерном классе 427А 10 корпуса ТПУ.

5.2. Правовые и организационные вопросы обеспечения безопасности

Разработка программного обеспечения происходит за компьютерным столом. Рабочее место должно удовлетворять требованиям ГОСТ 12.2.032-78 «Система стандартов безопасности труда (ССБТ). Рабочее место при выполнении работ сидя» [23] РД 153-34.0-03.298-2001 «Типовая конструкция по охране труда для пользователей персональными электронно-вычислительными машинами (ПЭВМ) в электроэнергетике» [24]. Требования к нормам труда (продолжительность рабочего дня, перерывы в течение рабочего дня, перерывы на обед) регламентируются ТК РФ «Рабочее время» [25].

При разработке программного обеспечения для формирования оптимального графика ППР было предоставлено рабочее место, где соблюдены все требования по организации труда с ЭВМ.

5.3. Производственная безопасность

При разработке программного обеспечения разработчики подвергаются воздействию различных вредных и опасных факторов, которые представлены в таблице 5.1. В таблице также представлены соответствующие нормативные документы и этапы работ, во время которых разработчики могут столкнуться с их влиянием.

Таблица 5.1 – Возможные опасные и вредные факторы

Факторы (ГОСТ 12.0.003-2015)	Этапы работ		Нормативные документы
	Разработка	Тестирование	
Отклонение показателей микроклимата	+	+	СанПиН 2.2.4.548-96 «Гигиенические требования к микроклимату производственных помещений» [26]
Недостаточная освещённость рабочей зоны	+	+	СП 52.13330.2016 «Естественное и искусственное освещение» [27]
Повышенная световая и цветовая контрастность	+	+	СП 52.13330.2016 «Естественное и искусственное освещение» [27]
Повышенный уровень шума на рабочем месте	+	+	СН 2.2.4/2.1.8.562-96 «Шум на рабочих местах, в помещениях жилых, общественных зданий и нам территории жилой застройки» [28]
Повышенный уровень статического электричества	+	+	ГОСТ Р 53734.1-2014 «Электростатические явления» [29]
Повышенная запылённость воздуха рабочей зоны	+	+	ГОСТ 12.1.005-88 «Общие санитарно-гигиенические требования к воздуху рабочей зоны» [30]
Опасность поражения электрическим током	+	+	ГОСТ Р 58698-2019 «Защита от поражения электрическим током» [31]

По данной таблице можно сделать вывод, что на разработчиков программного обеспечения в ходе их деятельности воздействуют только физические и психологические факторы, а химические и биологические факторы отсутствуют.

5.3.1. Отклонение показателей микроклимата

Отклонение показателей микроклимата на рабочем месте от комфортных непосредственно влияет на здоровье работников. Повышение скорости движения воздуха и понижение температуры может привести к переохлаждению организма путем усиления теплообмена и процесса теплоотдачи при испарении пота. Недостаточная влажность в свою очередь ведет к интенсивному испарению влаги со слизистых оболочек. Это может привести к пересыханию, растрескиванию и затем к заражению болезнетворными бактериями. При разработке программного обеспечения используются персональные компьютеры, которые могут непосредственно влиять на микроклимат путем снижения относительной влажности и повышению температуры в рабочем помещении.

Общие требования к микроклимату производственных помещений регламентируются СанПиН 2.2.4.548-96 «Гигиенические требования к микроклимату производственных помещений». Санитарные нормы регулируют оптимальные и допустимые значения показателей в рабочей зоне, соответствующие физиологическим потребностям организма человека, для создания комфортных и безопасных условий труда.

Работа, выполняемая командой разработки программного обеспечения, по энергозатратам относится к категории Iа (производится сидя, сопровождается незначительными физическими усилиями). В таблицах 5.2 и 5.3 представлены оптимальные и допустимые значения показателей микроклимата на рабочих местах для данной категории.

Таблица 5.2 – Оптимальные величины показателей микроклимата на рабочих местах

Период года	Температура воздуха, °С	Температура поверхностей, °С	Относительная влажность воздуха, %	Скорость движения воздуха, м/с
Холодный	22-24	21-25	60-40	0,1
Тёплый	23-25	22-26	60-40	0,1

Таблица 5.3 – Допустимые величины показателей микроклимата на рабочих местах

Период года	Температура воздуха, °С		Температура поверхностей, °С	Относительная влажность воздуха, %	Скорость движения воздуха, м/с	
	диапазон ниже оптимальных величин	диапазон выше оптимальных величин			для диапазона температур воздуха ниже оптимальных величин, не более	для диапазона температур воздуха выше оптимальных величин, не более
Холодный	20,0 – 21,9	24,1 – 25,0	19,0 – 26,0	15 – 75	0,1	0,1
Тёплый	21,0 – 22,9	25,1 – 28,0	20,0 – 29,0	15 – 75	0,1	0,1

В производственных помещениях, где поддерживать допускаемые нормативные величины локального микроклимата не представляется возможным, необходимо проводить мероприятия по защите работников от возможного перегревания и охлаждения. Это достигается разными способами: использование систем местного кондиционирования воздуха; регламентацией периодов работы в неблагоприятном локальном микроклимате и отдыха в помещении с микроклиматом, нормализующим тепловое состояние; уменьшение длительности рабочей смены и др.

5.3.2. Недостаточная освещенность рабочей зоны

Недостаточная освещенность рабочей зоны является вредным производственным фактором, приводящим к повышенной утомляемости и снижению работоспособности человека на предприятии. Продолжительная работа в условиях низкой освещенности приводит к ухудшению зрения.

Нормы естественного, искусственного и совместного освещения регламентируются СП 52.13330.2016 «Естественное и искусственное освещение». Разработка программного обеспечения относится к категории работ высокой точности – Б (наименьший или эквивалентный объект различения 0,30

– 0,50 мм), подразряд 1 (относительная продолжительность зрительной работы при направлении зрения на рабочую поверхность не менее 70%).

В таблице 5.4 представлены требования к освещению рабочего помещения для разряда Б1.

Таблица 5.4 – Требования к освещению рабочего помещения

Искусственное освещение				Естественное освещение	
Освещенность на рабочей поверхности от системы общего освещения, лк	Цилиндрическая освещенность	Объединенный показатель дискомфорта, не более	Коэффициент пульсации освещенности, Кп, %, не более	Коэффициент естественной освещенности, %, при	
				верхнем или комбинированном	боковом
300	100	21	15	3	1

Яркий свет в зоне периферийного зрения заметно увеличивает глазное напряжение. Для снижения влияния вредного фактора недостаточной освещенности необходимо, чтобы уровень естественного освещения рабочего пространства приблизительно совпадал с яркостью дисплея. Проблему недостаточной освещенности помещения можно решить при помощи установки дополнительных осветительных приборов, расширения световых проемов.

5.3.3. Повышенная световая и цветовая контрастность

Отклонение светового и цветового контраста на рабочем месте приводит к быстрому утомлению и снижению уровня работоспособности человека на предприятии. Продолжительное воздействие этого вредного фактора приводит к возникновению проблем со зрением. Нормы светового и цветового контраста регламентируются СП 52.13330.2016 «Естественное и искусственное освещение». Для работы за компьютером (категория работ Б1) нормы контраста представлены в таблице 5.5.

Таблица 5.5 – Требования к освещению рабочего помещения

Характеристика зрительной работы	Контраст объекта с фоном	Характеристика фона
Высокой точности	Малый	Средний
	Средний	Темный

Для изменения светового и цветового контраста необходимо отрегулировать уровень естественной и искусственной освещенности рабочего помещения или заменить текущее оборудование (мониторы) на более качественные, которые позволят сгладить контраст.

5.3.4. Повышенный уровень шума на рабочем месте

Превышение уровня шума на рабочем месте создает психологический и физический стресс, снижающий производительность, концентрацию, внимание, повышает утомляемость. Повышение уровня шума на рабочем месте возможно из-за фона, создаваемого работой персональных компьютеров, наличия центральной системы вентиляции и кондиционирования воздуха.

Предельно допустимые показатели уровня звука, звукового давления регламентируются СН 2.2.4/2.1.8.562-96 «Шум на рабочих местах, в помещениях жилых, общественных зданий и на территории жилой застройки»

Для команды разработчиков программного обеспечения, эти показатели представлены в таблице 5.6.

Таблица 5.6 – Предельно допустимые уровни звукового давления, уровни звука и эквивалентные уровни звука для инженера-программиста

Вид трудовой деятельности, рабочее место	Уровни звукового давления, дБ в октавных полосах со среднегеометрическими частотами, Гц						
	31,5	63	125	250	500	1000	2000
Творческая деятельность, руководящая работа с повышенными требованиями, научная деятельность, конструирование и проектирование, программирование, преподавание и обучение, врачебная деятельность. Рабочие места в помещениях дирекции, проектноконструкторских бюро, расчетчиков, программистов вычислительных машин, в лабораториях для теоретических работ и обработки данных, приема больных в здравпунктах	86	71	61	54	49	45	42

Для снижения уровня шума в производственном помещении можно использовать защитные звукопоглощающие экраны. Для любого оборудования необходимо регулярно проводить техническое обслуживание, так как загрязнение может увеличить производимый шум.

5.3.5. Повышенный уровень статического электричества

Статическое электричество является опасным производственным фактором, проявление которого может нанести вред здоровью человека (ожоги) или привести пожару и другим чрезвычайным ситуациям.

При работе за компьютером статический заряд может накапливаться, если нет хорошего контакта с землей или влажность/ионизация воздуха превышает допустимые нормы. Статический разряд в производственных помещениях

рассматриваемого типа при условии соответствии нормам микроклимата и организации работ при воздействии на человека вызывает дискомфорт.

Допустимые показатели уровня статического электричества на производстве регламентируются ГОСТ Р 53734.1-2014 «Электростатические явления». В таблице 5.7 представлены уровни восприятия электростатического заряда человеком.

Таблица 5.7 – Уровни восприятия людьми электростатического заряда и ответной реакции при емкости тела в 200 пФ

Энергия разряда, мДж	Реакция	Потенциал тела, В
0,1	Разряд ощутим	1000
0,9	Четко ощутим	3000
6,4	Неприятный шок	8000

Для уменьшения накопляемого статического заряда при работе за компьютером необходимо организовать антистатические рабочие места, соблюдать установленную норму влажности воздуха и поддерживать чистоту помещения, поскольку пыль обладает свойствами диэлектрика.

5.3.6. Повышенная запыленность воздуха рабочей зоны

Пыль характеризуется совокупностью свойств, определяющих поведение ее в воздухе, превращение и действие на организм человека.

Вредное воздействие пыли на организм человека зависит от ряда факторов: концентрации в воздухе, химического состава, размеров частиц, дисперсности, твердости, заряженности пылинок. Норма запыленности воздуха регламентируется ГОСТ 12.1.005-88 «Общие санитарно-гигиенические требования к воздуху рабочей зоны».

В офисном помещении, пыль может оказывать на организм раздражающее и аллергическое действия. В таблице 5.8 представлены предельно допустимые значения концентрации пыли и аэрозолей в воздухе жилых помещений.

Таблица 5.8 – Предельно допустимые концентрации пыли и аэрозолей в воздухе

Взвешенные частицы PM2.5	0,16 мг/м^3	0,035 мг/м^3	0,025 мг/м^3
Взвешенные частицы PM10	0,3 мг/м^3	0,06 мг/м^3	0,04 мг/м^3
Взвешенные частицы (общая пыль)	0,5 мг/м^3	0,15 мг/м^3	–
Сажа (углерод)	0,15 мг/м^3	0,05 мг/м^3	–

Для снижения уровня содержащейся в воздухе пыли необходимо организовать систему вентиляции воздуха помещения и производить регулярную уборку помещений.

5.3.7. Опасность поражения электрическим током

Под электробезопасностью подразумевается система технических и организационных мероприятий, направленных на защиту людей от опасного воздействия электрического тока, статического электричества и электромагнитного поля. Значения вышеперечисленных факторов регулируются ГОСТ Р 58698-2019.

Таблица 5.9 – Пороги напряжения прикосновения для реагирования

Характер реагирования	Пороги напряжения, В
Реакция испуга	2 (переменный ток)
	8 (постоянный ток)
Мышечная реакция	20 (переменный ток)
	40 (постоянный ток)

Меры предосторожности для основной защиты от поражения электрическим током:

- использование защитных ограждений или оболочек;
- размещение опасных для жизни и здоровья человека участков электропроводов и приборов вне зоны досягаемости рукой;
- ограничение напряжения или питание должно осуществляться от безопасного источника питания;

– автоматическое отключение питания (защитное устройство, которое будет отключать систему, питающую электрическое оборудование в случае замыкания)

Защита от поражения электрическим током может осуществляться посредством системы безопасного сверхнизкого напряжения (БСНН) и защитного сверхнизкого напряжения (ЗСНН).

5.4. Экологическая безопасность

Программное обеспечение не оказывает влияния на окружающую среду, так как его разработка и использование происходит при помощи персональных компьютеров, однако использование самого компьютера может оказывать влияние на окружающую среду. Так в случаях нагрева материнской платы и корпуса монитора происходит выброс в воздух вредных веществ, а в процессе работы компьютера, воздух вблизи него ионизируется, что приводит к повышенной сухости воздуха.

В производстве компьютеров и их комплектующих используются материалы, которые при неправильной утилизации компьютерной техники могут стать причиной загрязнения литосферы. Утилизировать компьютер необходимо после извлечения его компонент, их сортировки и отправки на повторное использование, это необходимо делать на специально отведённых полигонах с присутствием квалифицированного персонала. Стоит также учитывать, что в технологических процессах производства компьютеров и их комплектующих образуются производственные сточные воды, которые могут являться фактором загрязнения гидросферы.

Соблюдение всех норм при использовании и утилизации компьютерной техники позволяет уменьшить вредное воздействие на окружающую среду.

5.5. Безопасность в чрезвычайных ситуациях

5.5.1. Затопление

Главная опасность при затоплении помещения, в котором находятся ПК – это способность воды проводить электрический ток, что означает возможность поражения электрическим током человека, находящегося в таком помещении.

Ток проводят не сами молекулы воды, а различные примеси, содержащиеся в ней, такие как ионы различных минеральных солей, которые в достаточных количествах содержат сточные воды.

Затопление может иметь характер техногенной чрезвычайной ситуации, когда возникает по причине наличия сильной изношенности водопровода, свищей, негерметичных соединений водопроводных систем или в следствии аварийной ситуации. Также затопление может являться чрезвычайной ситуацией природного характера, в случаях, когда оно возникает в результате наводнений, паводков и т.д.

5.5.2. Землетрясение

Землетрясение – это подземные толчки и колебания земной поверхности из-за внезапных смещений и разрывов в земной коре или верхней мантии Земли, которые передаются на большие расстояния. Данная чрезвычайная ситуация имеет природный характер, может привести к выходу из строя коммуникаций и энергетических объектов, разрушению зданий, появлению трещин в грунте, возникновению пожаров, значительным людским потерям.

5.5.3. Короткое замыкание

Работа с персональными компьютерами подразумевает постоянное использование электрического тока. При несоблюдении правил электробезопасности возможно возникновение короткого замыкания проводки – резкое и многократное возрастание силы тока, протекающего в цепи, что приводит к значительному тепловыделению, расплавлению электрических проводов с последующим возникновением возгорания. Причиной короткого замыкания является нарушение изоляции и соединения токопроводящих частей электроустановок друг с другом или с заземлёнными поверхностями непосредственно или через токопроводящий материал. К нарушениям изоляции могут привести перенапряжение, прямые удары молнии, внешние механические повреждения, старение и износ самой изоляции, в том числе возникшие из-за неудовлетворительного ухода.

Если человек находится рядом с участком цепи в котором произошло короткое замыкание, он может получить ожоги, в том числе смертельные. Компьютеры, подключённые в цепь, в которой произошло короткое замыкание могут выйти из строя. Для минимизации перечисленных негативных последствий короткого замыкания следует использовать кабель не распространяющий горение, или помещать кабель в стальные трубы с определённой толщиной стенки, которая не прожжётся при возникновении короткого замыкания.

5.5.4. Пожар

Причинами возникновения пожара при работе с ПК может служить короткое замыкание проводки, в том числе в следствии неисправности прибора, сильный перегрев ПК в результате его использования в режиме повышенной нагрузки.

Для предотвращения возникновения пожара, необходимо проводить периодическую своевременную диагностику оборудования и электрической проводки, соблюдать нормы при работе с ПК, обеспечить наличие средств пожаротушения в рабочем помещении, готовых к эксплуатации. Здание, в котором находится помещение с ПК, тоже должно отвечать требованиям пожарной безопасности, для этого необходимо наличие охранно-пожарной сигнализации, плана эвакуации, углекислотных огнетушителей с проверенным клеймом, табличек с указанием направления к эвакуационному выходу. При появлении возгорания необходимо сообщить в службу пожарной охраны адрес и место возникновения пожара.

5.6. Вывод по разделу

В результате работы над разделом «Социальная ответственность» были выявлены основные нормативные акты для обеспечения безопасности жизнедеятельности на рабочем месте, рассмотрены наиболее значимые опасные и вредные факторы, возникающие при разработке программного обеспечения для формирования графика планово-предупредительных ремонтных работ. Приведены меры, необходимые для снижения влияния возникающих опасных и вредных факторов на организм человека. Описано влияние процесса разработки данного программного обеспечения на окружающую среду и указаны меры, необходимые для сокращения негативного влияния. Выявлены возможные для описываемого рабочего места чрезвычайные ситуации, приведён ряд действий, необходимых для предотвращения наиболее типичной ЧС.

Заключение

В результате выполнения выпускной квалификационной работы была разработана интеллектуальная система для прогнозирования даты проведения планово-предупредительных ремонтных работ на производственном оборудовании.

В качестве одного из компонентов интеллектуальной системы разработана программная реализация нейро-нечёткой сети для прогнозирования коэффициента неравномерности загрузки оборудования. Реализация написана на языке программирования Python с использованием таких библиотек как PyTorch, NumPy, Matplotlib. Структура разработанной сети включает в себя ANFIS модель, состоящую из пяти слоёв, трёх входов и одного выхода, а также линейную сеть с одним входом и выходом, имеющую настраиваемое количество скрытых слоёв. Также реализована возможность пользовательской настройки таких параметров как количество входов, скрытых слов и нейронов на них, эпох; типы функций активаций и алгоритма логического вывода ANFIS; число и вид лингвистических правил; тип и количество функций принадлежности. Приведены рекомендации по выбору параметров настройки ННС. По результатам вычислительных экспериментов ошибка прогнозирования не превышала 5%. Разработан алгоритм вычисления остаточного ресурса и необходимого запаса сменных деталей на основании коэффициента неравномерности загрузки.

Далее, был разработан алгоритм формирования оптимального графика ППР с минимизацией суммарной остаточной стоимости оборудования и стоимости простоя.

Реализован программный продукт с графическим интерфейсом пользователя, который позволяет управлять всеми вышеперечисленными компонентами созданной интеллектуальной системы.

Список публикаций студентов

1. Ильина С. А. Разработка конфигуратора прогноза планово-предупредительных ремонтов с применением нейро-нечётких систем / С. А. Ильина, М. Е. Семенов // Материалы докладов XVII Международной научно-практической конференции «Электронные средства и системы управления». – 2021. - ч. 2. - С. 308-310.
2. Semenov, M. Neural fuzzy network configurator for calculating a residual life of production equipment / M. Semenov, S. Ilina, A. Rutskov // Journal of Physics: Conference Series, 2022, 1989(1), 012011 (в печати)
3. Ильина С.А. Разработка конфигуратора нейро-нечёткой сети для прогнозирования коэффициента загрузки оборудования // Сборник научных трудов XIX Международной конференции студентов, аспирантов и молодых ученых "Перспективы развития фундаментальных наук", 26–29 апреля 2022 г., Россия, Томск (в печати).
4. Исмагилов Р.Р. Алгоритм формирования оптимального графика планово-предупредительных ремонтных работ / Р.Р. Исмагилов, Е.А. Мироненко, А.Л. Руцков, М.Е. Семенов // Сборник научных трудов XIX Международной конференции студентов, аспирантов и молодых ученых "Перспективы развития фундаментальных наук", 26–29 апреля 2022 г., Россия, Томск (в печати).

Список использованных источников

1. Digital Design [Электронный ресурс]. – Режим доступа: https://digdes.ru/, свободный (дата обращения: 13.06.2022)
2. Баскакова Н.Т., Дорман В.Н. К вопросу эффективности аутсорсинга ремонтных работ на промышленном предприятии // Экономический анализ: теория и практика. – 2017, т. 16, вып. 2. – С. 351–363
3. Lei Y. Intelligent Fault Diagnosis and Remaining Useful Life Prediction of Rotating Machinery / Y. Lei ; Butterworth-Heinemann, 2017. – 366 p.
4. Сай Ван Квонг Модели и методы проактивной поддержки принятия решений при управлении техническим состоянием оборудования: дис. канд. тех. наук / Волгоградский государственный технический университет – Волгоград, 2020. – 152 с.
5. Angelov P. Simplified fuzzy rule-based systems using non-parametric antecedents and relative data density / P. Angelov, R. Yager // 2011 IEEE Workshop on Evolving and Adaptive Intelligent Systems (EAIS). – 2011. – pp. 62–69
6. Bahman Najafi, Sina Faizollahzadeh Ardabili Application of ANFIS, ANN, and logistic methods in estimating biogas production from spent mushroom compost (SMC) // Resources, Conservation and Recycling, Volume 133, June 2018, Pages 169-178
7. Нижневский В. В., Матвеев М. Г. Реализация алгоритмов нечёткого логического вывода на языке программирования Python / Сборник студенческих научных работ факультета компьютерных наук ВГУ – 2019. – С. 152-157
8. Солдатова, О. П. Алгоритм минимизации базы правил нечеткой нейронной сети Такаги-Сугено-Канга / О. П. Солдатова, Ю. М. Шепелев // European research: сборник статей победителей X Международной научно-практической конференции (Пенза: "Наука и Просвещение"). – 2017. – С. 46–49.

9. Осовский С. Нейронные сети для обработки информации / Пер. с польского И.Д. Рудинского. – М.: Финансы и статистика, 2002. – 344 с.
10. В. Дьяконов, В. Круглов. Математические пакеты расширения MATLAB. Специальный справочник. -Санкт-Петербург: Питер, 2001 - с. 307-309.
11. Леоненков А. В. Нечеткое моделирование в среде MATLAB и fuzzyTECH. – СПб.: БХВ Петербург, 2005. – 736 с.
12. ANFIS in pyTorch [Электронный ресурс]. – Режим доступа: https://github.com/jfpower/anfis-pytorch/, свободный (дата обращения: 23.07.2021)
13. Jang J S 1993 IEEE Transactions on Systems, Man, and Cybernetics 23 665–685
14. Pereira J. Procedures for the bin packing problem with precedence constraints. European Journal of Operational Research. 2016. – 250(3). – pp. 794-806.
15. Фуремс Е.М. Обратная задача об упаковке в контейнеры при наличии качественных критериев – постановка и обзор применяемых методов // Искусственный интеллект и принятие решений. – 2016. № 3. – С. 31–43.
16. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированного проектирования. Паттерны проектирования. — СПб: Питер, 2001. — 368 с.: ил. (Серия «Библиотека программиста»)
17. Хомоненко А.Д., Цыганков В.М., Мальцев М.Г. - Базы данных. Учебник для высших учебных заведений (6-е изд.). 2009
18. Рыбанов А.А., Свиридова О.В., Короткова Н.Н., Лясин Д.Н., Абрамова О.Ф., Модель оценки сложности физической схемы реляционной базы данных // Инженерный вестник Дона. 2019. №3.

19. Проектирование интерфейсов пользователя: пособие для студентов специальности 1-47 01 02 «Дизайн электронных и веб-изданий» / Т. П. Брусенцова, Т. В. Кишкурно. – Минск : БГТУ, 2019. – 172 с.
20. Мухаметгалиева К. А. Патентование программного обеспечения / К. А. Мухаметгалиева // Правовая защита, экономика и управление интеллектуальной собственностью : материалы всероссийской научно-практической конференции, Екатеринбург, 21 апреля 2015 г. — Екатеринбург : [УрФУ], 2015. — Т. 1 — С. 84-91.
21. Справочник организаций и предприятий России [Электронный ресурс]. – Режим доступа: https://sfo.spr.ru/, свободный (дата обращения 27.05.2022)
22. Content AI [Электронный ресурс]. – Режим доступа: https://contentai.ru/, свободный (дата обращения 27.05.2022)
23. ГОСТ 12.2.032-78 Система стандартов безопасности труда (ССБТ). Рабочее место при выполнении работ сидя. Общие эргономические требования // Электронный фонд правовой и нормативно-технической документации [Электронный ресурс]. – Режим доступа: https://docs.cntd.ru/document/1200003913 (дата обращения: 12.05.2022).
24. РД 153-34.0-03.298-2001 Типовая инструкция по охране труда для пользователей персональными электронно-вычислительными машинами (ПЭВМ) в электроэнергетике // Электронный фонд правовой и нормативно-технической документации [Электронный ресурс]. – Режим доступа: https://docs.cntd.ru/document/1200031404 (дата обращения: 12.05.2022).
25. Трудовой кодекс (ТК РФ) «Рабочее время»// Электронный фонд правовой и нормативно-технической документации [Электронный ресурс]. – Режим доступа: http://base.garant.ru/12125268/ (дата обращения: 12.05.2022).
26. СанПиН 2.2.4.548-96 Гигиенические требования к микроклимату производственных помещений // Электронный фонд правовой и

нормативно-технической документации [Электронный ресурс]. 2021. – Режим доступа: https://docs.cntd.ru/document/901704046 (дата обращения: 12.05.2022).

27. СП 52.13330.2016 «Естественное и искусственное освещение» // Электронный фонд правовой и нормативно-технической документации [Электронный ресурс]. – Режим доступа: https://docs.cntd.ru/document/456054197 (дата обращения: 12.05.2022).
28. СН 2.2.4/2.1.8.562-96 «Шум на рабочих местах, в помещениях жилых, общественных зданий и на территории жилой застройки» // Электронный фонд правовой и нормативно-технической документации [Электронный ресурс]. – Режим доступа: https://docs.cntd.ru/document/901703278 (дата обращения: 12.05.2022).
29. ГОСТ Р 53734.1-2014 «Электростатические явления» // Электронный фонд правовой и нормативно-технической документации [Электронный ресурс]. – Режим доступа: https://docs.cntd.ru/document/1200111323 (дата обращения: 12.05.2022).
30. ГОСТ 12.1.005-88 «Общие санитарно-гигиенические требования к воздуху рабочей зоны» // Электронный фонд правовой и нормативнотехнической документации [Электронный ресурс]. – Режим доступа: https://docs.cntd.ru/document/1200003608 (дата обращения: 12.05.2022).
31. ГОСТ Р 58698-2019 «Защита от поражения электрическим током» // Электронный фонд правовой и нормативнотехнической документации [Электронный ресурс]. – Режим доступа: https://docs.cntd.ru/document/1200170001 (дата обращения: 12.05.2022).

Приложение А

Таблица А.1 – Таблица обучения ННС по сгруппированным k неравномерности

Показатели	**k неравномерности по SCADA**	**k неравномерности по опер. отчётам**	**k неравномерности по имит. модели**	**k неравномерности ННС**
Параметр	Взвешенная сумма по всем датчикам SCADA	Взвешенная сумма по всем оперативным отчётам	Взвешенная сумма по имитационной модели	Итоговый коэффициент неравномерности
1	**2**	**3**	**4**	**5**
Значения	0	0	0	0
	0.05	0	0	0.05
	0.05	0.05	0.05	0.06
	0.1	0.05	0.05	0.11
	0.1	0.1	0.1	0.12
	0.15	0.1	0.1	0.17
	0.15	0.15	0.15	0.18
	0.2	0.15	0.15	0.23
	0.25	0.2	0.2	0.29
	0.25	0.25	0.25	0.3
	0.3	0.25	0.25	0.35
	0.3	0.3	0.3	0.36
	0.35	0.3	0.3	0.41
	0.35	0.35	0.35	0.42
	0.45	0.35	0.35	0.52
	0.45	0.4	0.4	0.53
	0.45	0.45	0.45	0.54
	0.5	0.45	0.45	0.59
	0.50	0.50	0.50	0.60
	0.55	0.50	0.50	0.65
	0.55	0.60	0.60	0.68

Продолжение таблицы А.1

1	**2**	**3**	**4**	**5**
	0.60	0.60	0.60	0.70
	0.60	0.65	0.65	0.72
	0.65	0.65	0.65	0.75
	0.65	0.70	0.70	0.77
	0.65	0.75	0.75	0.78
	0.70	0.75	0.75	0.80
	0.70	0.80	0.80	0.82
	0.75	0.80	0.80	0.85
	0.75	0.85	0.85	0.87
	0.80	0.85	0.85	0.89
	0.80	0.90	0.90	0.90
	0.80	0.95	0.95	0.92
	0.85	0.95	0.95	0.95
	0.85	1.00	1.00	0.97
	0.90	1.00	1.00	0.99
	0.90	1.05	1.05	1.00
	0.95	1.05	1.05	1.03
	0.95	1.10	1.10	1.05
	1.00	1.10	1.10	1.08
	1.00	1.15	1.15	1.10
	1.50	1.50	1.50	1.78
	2.00	2.20	2.20	2.50
	2.50	2.50	2.50	2.70

Таблица А.2 – Тестовый набор данных

k неравномерности по SCADA	**k неравномерности по опер. отчётам**	**k неравномерности по имит. модели**	**k неравномерности полученные ННС ННС**
1	**2**	**3**	**4**
0.05	0.05	0.05	0.06
0.10	0.10	0.10	0.12
0.15	0.15	0.15	0.18
0.20	0.20	0.20	0.29
0.25	0.25	0.25	0.30
0.30	0.30	0.30	0.36
0.35	0.35	0.35	0.42
0.40	0.40	0.40	0.48
0.45	0.45	0.45	0.54
0.50	0.50	0.50	0.60
0.55	0.55	0.55	0.67
0.60	0.60	0.60	0.70
0.65	0.65	0.65	0.75
0.70	0.70	0.70	0.78
0.75	0.75	0.75	0.81
0.80	0.80	0.80	0.89
0.85	0.85	0.85	0.90
0.90	0.90	0.90	0.95
0.95	0.95	0.95	1.02
1.00	1.00	1.00	1.06
1.05	1.05	1.05	1.07
1.10	1.10	1.10	1.09
1.15	1.15	1.15	1.11

Продолжение таблицы А.2

1	**2**	**3**	**4**
1.20	1.20	1.20	1.15
1.25	1.25	1.25	1.30
1.30	1.30	1.30	1.40
1.35	1.35	1.35	1.50
1.40	1.40	1.40	1.60
1.45	1.45	1.45	1.70
1.50	1.50	1.50	1.78
1.55	1.55	1.55	1.84
1.60	1.60	1.60	1.90
1.65	1.65	1.65	1.97
1.70	1.70	1.70	2.00
1.75	1.75	1.75	2.05
1.80	1.80	1.80	2.10
1.85	1.85	1.85	2.15
1.90	1.90	1.90	2.18
1.95	1.95	1.95	2.20
2.00	2.00	2.00	2.25
2.05	2.05	2.05	2.30
2.10	2.10	2.10	2.35
2.15	2.15	2.15	2.40
2.20	2.20	2.20	2.52
2.25	2.25	2.25	2.53
2.30	2.30	2.30	2.57
2.35	2.35	2.35	2.60
2.40	2.40	2.40	2.64
2.45	2.45	2.45	2.67
2.50	2.50	2.50	2.70

Продолжение таблицы А.2

1	**2**	**3**	**4**
2.55	2.55	2.55	2.74
2.60	2.60	2.60	2.77
2.65	2.65	2.65	2.80
2.70	2.70	2.70	2.83
2.75	2.75	2.75	2.87
2.80	2.80	2.80	2.90
2.85	2.85	2.85	2.93
2.90	2.90	2.90	2.97
2.95	2.95	2.95	3.00
3.00	3.00	3.00	3.03
3.05	3.05	3.05	3.07
3.10	3.10	3.10	3.10
3.15	3.15	3.15	3.13
3.20	3.20	3.20	3.16
3.25	3.25	3.25	3.20
3.30	3.30	3.30	3.23
3.35	3.35	3.35	3.26
3.40	3.40	3.40	3.30
3.45	3.45	3.45	3.33
3.50	3.50	3.50	3.36
3.55	3.55	3.55	3.40
3.60	3.60	3.60	3.43
3.65	3.65	3.65	3.46
3.70	3.70	3.70	3.49
0.05	0.05	0.05	0.06

Таблица А.3 – Пример производственных данных

Дата и время	**WICA_1301**	**FIC_VRP1301_VAL**	**VRPT1301_VAL**
01.04.2020 0:00	9250	7,23379612	7,23379612
01.04.2020 0:01	9700	7,23379612	7,23379612
01.04.2020 0:02	10140	7,23379612	7,23379612
01.04.2020 0:03	10580	7,23379612	7,23379612
01.04.2020 0:04	11030	7,23379612	7,23379612
01.04.2020 0:05	11470	7,23379612	7,23379612
01.04.2020 0:06	11910	7,23379612	7,23379612
01.04.2020 0:07	12350	7,23379612	7,23379612
01.04.2020 0:08	12790	0	0
01.04.2020 0:09	13230	0	7,23379612
01.04.2020 0:11	14100	7,23379612	7,23379612
01.04.2020 0:12	14540	7,23379612	7,23379612
01.04.2020 0:13	14970	0	0
01.04.2020 0:14	15410	0	0
01.04.2020 0:15	15840	7,23379612	7,23379612
01.04.2020 0:16	16270	7,23379612	7,23379612
01.04.2020 0:17	16700	7,23379612	7,23379612
01.04.2020 0:18	17120	7,23379612	7,23379612
01.04.2020 0:19	17550	7,23379612	7,23379612
01.04.2020 0:20	17970	7,23379612	7,23379612
01.04.2020 0:21	18400	0	0
01.04.2020 0:22	18960	7,23379612	7,23379612
01.04.2020 0:23	19000	0	0
01.04.2020 0:24	19000	7,23379612	7,23379612
01.04.2020 0:28	20000	6611,689941	6611,689941
01.04.2020 0:29	19730	19820,60156	19741,0293
01.04.2020 0:30	19450	19878,47266	19878,47266
01.04.2020 0:31	19160	19907,4082	19907,4082
01.04.2020 0:32	18870	19755,49805	19892,93945
01.04.2020 0:33	18590	20167,82422	20052,08398

Приложение Б

Структура ANFIS части ННС.

```
import itertools
from collections import OrderedDict
import numpy as np
import torch
import torch.nn.functional as F
dtype = torch.float

class FuzzifyVariable(torch.nn.Module):
    '''
        Класс описывает одну нечёткую переменную
        содержит список её функций принадлежности
    '''
    #инициализация
    def __init__(self, mfdefs):
        super(FuzzifyVariable, self).__init__()
        if isinstance(mfdefs, list):  # No MF names supplied
            mfnames = ['mf{}'.format(i) for i in range(len(mfdefs))]
            mfdefs = OrderedDict(zip(mfnames, mfdefs))
        self.mfdefs = torch.nn.ModuleDict(mfdefs)
        self.padding = 0

    @property
    def num_mfs(self):
        '''Возвращает фактическое число функций принадлежности '''
        return len(self.mfdefs)

    def members(self):
        '''
            Возвращает итератор по функциям принадлежности переменной
            в виде кортежей (имя_фп, MembFunc-объект)
        '''
        return self.mfdefs.items()

    def pad_to(self, new_size):
        '''
            Заполнение результата прямого прохода нулями, чтобы он имел
нужный размер
        '''
        self.padding = new_size - len(self.mfdefs)

    def fuzzify(self, x):
```

```
        '''
            Выводит список (имя_фп, нечёткое значение в x) для данных входных
значений
        '''
        for mfname, mfdef in self.mfdefs.items():
            yvals = mfdef(x)
            yield(mfname, yvals)

    def forward(self, x):
        '''
            x.shape: n_cases
            y.shape: n_cases * n_mfs
            Возвращает тензор, дающий значения принадлежности для каждой фп
        '''
        y_pred = torch.cat([mf(x) for mf in self.mfdefs.values()], dim=1)
        if self.padding > 0:
            y_pred = torch.cat([y_pred,
                                torch.zeros(x.shape[0], self.padding)], dim=1)
        return y_pred

class FuzzifyLayer(torch.nn.Module):
    '''
        Класс представляет собой список нечётких переменных, составляющих
входной слой для ННС.
    '''
    def __init__(self, varmfs, varnames=None):
        super(FuzzifyLayer, self).__init__()
        if not varnames:
            self.varnames = ['x{}'.format(i) for i in range(len(varmfs))]
        else:
            self.varnames = list(varnames)
        maxmfs = max([var.num_mfs for var in varmfs])
        for var in varmfs:
            var.pad_to(maxmfs)
        self.varmfs = torch.nn.ModuleDict(zip(self.varnames, varmfs))

    @property
    def num_in(self):
        '''Возвращает число входных переменных '''
        return len(self.varmfs)

    @property
    def max_mfs(self):
        ''' Возвращает максимальное число функций принадлежности среди всех
```

```
переменных'''
        return max([var.num_mfs for var in self.varmfs.values()])

    def __repr__(self):
        '''
            Вывод переменных, функций принадлежности и их параметров
        '''
        r = ['Input variables']
        for varname, members in self.varmfs.items():
            r.append('Variable {}'.format(varname))
            for mfname, mfdef in members.mfdefs.items():
                r.append('- {}: {}({})'.format(mfname,
                        mfdef.__class__.__name__,
                        ', '.join(['{}={}'.format(n, p.item())
                                for n, p in mfdef.named_parameters()])))
        return '\n'.join(r)

    def forward(self, x):
        ''' x.shape = n_cases * n_in
            y.shape = n_cases * n_in * n_mfs
            Даёт нечёткое представление значений каждой переменной,
            используя соответствующие им функции принадлежности
        '''
        assert x.shape[1] == self.num_in,\
            '{} is wrong no. of input values'.format(self.num_in)
        y_pred = torch.stack([var(x[:,i:i+1])
                              for i, var in enumerate(self.varmfs.values())],
                             dim=1)
        return y_pred

class AntecedentLayer(torch.nn.Module):
    '''
        Формирует "правила" используя все возможные комбинации функций
принадлежности
        для каждой переменной
    '''
    def __init__(self, varlist):
        super(AntecedentLayer, self).__init__()
        mf_count = [var.num_mfs for var in varlist]
        mf_indices = itertools.product(*[range(n) for n in mf_count])
        self.mf_indices = torch.tensor(list(mf_indices))
        # mf_indices.shape is n_rules * n_in

    def num_rules(self):
```

```
        '''Возвращает число полученных "правил"'''
        return len(self.mf_indices)

    def print_rules(self):
        return self.mf_indices

    def extra_repr(self, varlist=None):
        if not varlist:
            return None
        row_ants = []
        mf_count = [len(fv.mfdefs) for fv in varlist.values()]
        for rule_idx in itertools.product(*[range(n) for n in mf_count]):
            thisrule = []
            for (varname, fv), i in zip(varlist.items(), rule_idx):
                thisrule.append('{} is {}'
                                .format(varname, list(fv.mfdefs.keys())[i]))
            row_ants.append(' and '.join(thisrule))
        return '\n'.join(row_ants)

    def forward(self, x):
        ''' Calculate the fire-strength for (the antecedent of) each rule
            x.shape = n_cases * n_in * n_mfs
            y.shape = n_cases * n_rules
        '''
        batch_indices = self.mf_indices.expand((x.shape[0], -1, -1))
        ants = torch.gather(x.transpose(1, 2), 1, batch_indices)
        rules = torch.prod(ants, dim=2)
        return rules

class ConsequentLayer(torch.nn.Module):
    '''
        Простой линейный слой для представления Takagi–Sugeno–Kang следствий.
        Гибридное обучение с использованием среднеквадратичной ошибки для настройки коэффициентов.
    '''
    def __init__(self, d_in, d_rule, d_out):
        super(ConsequentLayer, self).__init__()
        c_shape = torch.Size([d_rule, d_out, d_in+1])
        self._coeff = torch.zeros(c_shape, dtype=dtype, requires_grad=True)

    @property
    def coeff(self):
        '''
```

```
        coeff.shape: n_rules * n_out * (n_in+1)
        Запись текущих коэффициентов для всех правил
    '''
    return self._coeff

@coeff.setter
def coeff(self, new_coeff):
    '''
        coeff: for each rule, for each output variable:
              a coefficient for each input variable, plus a constant
        Запись новых коэффициентов для всех правил
    '''
    assert new_coeff.shape == self.coeff.shape, \
        'Coeff shape should be {}, but is actually {}'\
        .format(self.coeff.shape, new_coeff.shape)
    self._coeff = new_coeff

def fit_coeff(self, x, weights, y_actual):
    '''
        Use LSE to solve for coeff: y_actual = coeff * (weighted)x
            x.shape: n_cases * n_in
        weights.shape: n_cases * n_rules
        [ coeff.shape: n_rules * n_out * (n_in+1) ]
            y.shape: n_cases * n_out
    '''
    x_plus = torch.cat([x, torch.ones(x.shape[0], 1)], dim=1)
    # Shape of weighted_x is n_cases * n_rules * (n_in+1)
    weighted_x = torch.einsum('bp, bq -> bpq', weights, x_plus)
    # Can't have value 0 for weights, or LSE won't work:
    weighted_x[weighted_x == 0] = 1e-12
    # Squash x and y down to 2D matrices for gels:
    weighted_x_2d = weighted_x.view(weighted_x.shape[0], -1)
    y_actual_2d = y_actual.view(y_actual.shape[0], -1)
    # Use gels to do LSE, then pick out the solution rows:
    try:
        coeff_2d = torch.linalg.lstsq(y_actual_2d, weighted_x_2d).solution
    except RuntimeError as e:
        print('Internal error in gels', e)
        print('Weights are:', weighted_x)
        raise e
    coeff_2d = coeff_2d[0:weighted_x_2d.shape[1]]
    # Reshape to 3D tensor: divide by rules, n_in+1, then swap last 2 dims
    self.coeff = coeff_2d.view(weights.shape[1], x.shape[1]+1, -1)\
        .transpose(1, 2)
```

```
        # coeff dim is thus: n_rules * n_out * (n_in+1)

    def forward(self, x):
        '''
            Calculate: y = coeff * x + const   [NB: no weights yet]
                x.shape: n_cases * n_in
             coeff.shape: n_rules * n_out * (n_in+1)
                y.shape: n_cases * n_out * n_rules
        '''
        # Append 1 to each list of input vals, for the constant term:
        x_plus = torch.cat([x, torch.ones(x.shape[0], 1)], dim=1)
        # Need to switch dimansion for the multipy, then switch back:
        y_pred = torch.matmul(self.coeff, x_plus.t())
        return y_pred.transpose(0, 2)  # swaps cases and rules

class PlainConsequentLayer(ConsequentLayer):
    '''
        Линейный слой для представления Takagi–Sugeno–Kang следствий.
        Не гибридное обучение, коэффициенты являются параметрами обратного
распространения.
    '''
    def __init__(self, *params):
        super(PlainConsequentLayer, self).__init__(*params)
        self.register_parameter('coefficients',
                                torch.nn.Parameter(self._coeff))

    @property
    def coeff(self):
        '''
            coeff.shape: n_rules * n_out * (n_in+1)
            Запись текущих коэффициентов для всех правил
        '''
        return self.coefficients

    def fit_coeff(self, x, weights, y_actual):
        '''
        '''
        assert False,\
            'Not hybrid learning: I\'m using BP to learn coefficients'

class WeightedSumLayer(torch.nn.Module):
    '''
        Сумматор
    '''
```

```
    def __init__(self):
        super(WeightedSumLayer, self).__init__()

    def forward(self, weights, tsk):
        '''
            weights.shape: n_cases * n_rules
               tsk.shape: n_cases * n_out * n_rules
             y_pred.shape: n_cases * n_out
        '''
        # Add a dimension to weights to get the bmm to work:
        y_pred = torch.bmm(tsk, weights.unsqueeze(2))
        return y_pred.squeeze(2)

class AnfisNet(torch.nn.Module):
    '''
        Класс ANFIS сети.
        Прямой проход сопоставляет входы с выходами на основе текущих
настроек
        после чего коэффициенты TSK настраиваются при помощи LSE
    '''
    def __init__(self, description, invardefs, outvarnames, hybrid=True):
        super(AnfisNet, self).__init__()
        self.description = description
        self.outvarnames = outvarnames
        self.hybrid = hybrid
        varnames = [v for v, _ in invardefs]
        mfdefs = [FuzzifyVariable(mfs) for _, mfs in invardefs]
        self.num_in = len(invardefs)
        self.num_rules = np.prod([len(mfs) for _, mfs in invardefs])
        if self.hybrid:
            cl = ConsequentLayer(self.num_in, int(self.num_rules), self.num_out)
        else:
            cl = PlainConsequentLayer(self.num_in, self.num_rules, self.num_out)
        self.layer = torch.nn.ModuleDict(OrderedDict([
            ('fuzzify', FuzzifyLayer(mfdefs, varnames)),
            ('rules', AntecedentLayer(mfdefs)),
            # normalisation layer is just implemented as a function.
            ('consequent', cl),
            ('weighted_sum', WeightedSumLayer())
            ]))

    @property
    def num_out(self):
        return len(self.outvarnames)
```

```
@property
def coeff(self):
    return self.layer['consequent'].coeff

@coeff.setter
def coeff(self, new_coeff):
    self.layer['consequent'].coeff = new_coeff

def fit_coeff(self, x, y_actual):
    '''
        Прямой проход для получения весов.
    '''
    if self.hybrid:
        self(x)
        self.layer['consequent'].fit_coeff(x, self.weights, y_actual)

def input_variables(self):
    '''
        Возвращает итератор по входным переменным системы.
        Выдаёт кортежи вида (имя_переменной, FuzzifyVariable-объект)
    '''
    return self.layer['fuzzify'].varmfs.items()

def output_variables(self):
    '''
        Возвращает список имён выходных переменных системы.
    '''
    return self.outvarnames

def extra_repr(self):
    rstr = []
    vardefs = self.layer['fuzzify'].varmfs
    rule_ants = self.layer['rules'].extra_repr(vardefs).split('\n')
    for i, crow in enumerate(self.layer['consequent'].coeff):
        rstr.append('Rule {:2d}: IF {}'.format(i, rule_ants[i]))
        rstr.append(' '*9+'THEN {}'.format(crow.tolist()))
    return '\n'.join(rstr)

def forward(self, x):
    '''
        Реализация прямого прохода.
        Выходные данные каждого слоя сохраняются в переменной экземпляра
    '''
```

```
        'first layer: input of the fuzzt system'
        self.fuzzified = self.layer['fuzzify'](x)
        #print('fuzzified', self.fuzzified)
        'second layer: MFs decide the fuzzy rules'
        self.raw_weights = self.layer['rules'](self.fuzzified)
        #print('rules', self.raw_weights)
        'third layer: normalizes'
        self.weights = F.normalize(self.raw_weights, p=1, dim=1)
        #print('weights', self.weights)
        'fourth layer: provides the first model by derived parameters'
        self.rule_tsk = self.layer['consequent'](x)
        'fifth layer: summator'
        y_pred = self.layer['weighted_sum'](self.weights, self.rule_tsk)
        print ('sum', y_pred)
        self.y_pred = y_pred
        return self.y_pred
```

Структура линейной части ННС.

```
import torch
from Model.ANFIS_2.anfis_net import plots
from Model.ANFIS_2.anfis_net import experimental
from torchsummary import summary

class LinearNet(torch.nn.Module):
    def __init__(self, n_hidden_neurons, n_hidden_layers=3, actF =
['Sigmoid','GELU','ReLU','Sigmoid']):
        super(LinearNet, self).__init__()
        self.fc1 = torch.nn.Linear(1, n_hidden_neurons)
        self.fc2 = torch.nn.Linear(n_hidden_neurons, n_hidden_neurons)
        self.fc3 = torch.nn.Linear(n_hidden_neurons, 1)
        self.actS = torch.nn.Sigmoid()
        self.actG = torch.nn.GELU()
        self.actR = torch.nn.ReLU()
        self.layers = n_hidden_layers
        self.actF = actF
    def forward(self, x):
        x = self.fc1(x)
        if self.actF[0] == 'Sigmoid':
            x = self.actS(x)
        if self.actF[0] == 'GELU':
            x = self.actG(x)
        if self.actF[0] == 'ReLU':
            x = self.actR(x)
```

```
        if self.layers == 2:
            x = self.fc3(x)
        else:
            for i in range(self.layers - 2):
                x = self.fc2(x)
                if self.actF[i+1] == 'Sigmoid':
                    x = self.actS(x)
                if self.actF[i+1] == 'GELU':
                    x = self.actG(x)
                if self.actF[i+1] == 'ReLU':
                    x = self.actR(x)
            x = self.fc3(x)
        print('lin', x)
        return x

def train(model, criterion, x, y_actual, show_plot=False, error_plot=False,
epoch=1000):
    errors = []  # для графика
    for param in model.parameters():
        param.requires_grad = True
    optimizer = torch.optim.Rprop(model.parameters(), lr=0.01)
    x = x.clone().detach().requires_grad_(True)
    for i in range(epoch):
        optimizer.zero_grad()
        y_pred = model.forward(x)
        loss_val = criterion(y_pred, y_actual)
        loss_val.backward()
        optimizer.step()
        mse, rmse, perc_loss = experimental.calc_error(y_pred, y_actual)
        errors.append(perc_loss)
        if perc_loss < 1.5:
            break
    summary(model, x.shape)
    if error_plot:
        plots.plotErrors(errors, "Ошибка НС")
    if show_plot:
        plots.plotResults(y_actual, y_pred, "Результаты НС")
    return y_pred, errors[epoch-2]

class TrainedNFS(torch.nn.Module):
    def __init__(self, anfis, linearNet):
        super(TrainedNFS, self).__init__()
        self.anfis = anfis
        self.linearNet = linearNet
```

```
    def forward(self, x):
        y_pred = self.anfis.forward(x)
        y_pred = self.linearNet.forward(y_pred)
        return y_pred
```

Обучение, тестирование, вычисление ошибок.

```
import torch
import torch.nn.functional as F
from Model.ANFIS_2.anfis_net import plots
from Model.ANFIS_2.anfis_net import linearNet
import matplotlib.pyplot as plt
dtype = torch.float

#функция для подсчёта ошибок
def calc_error(y_pred, y_actual):
    with torch.no_grad():
        tot_loss = F.mse_loss(y_pred, y_actual)
        rmse = torch.sqrt(tot_loss).item()

        perc_loss = torch.mean(100. * torch.abs((y_pred - y_actual)
                            / y_actual))
    return(tot_loss, rmse, perc_loss)

#тестирование НС
def test_anfis(model, data, show_plots=False):
    test_loss = 0
    x, y_actual = data.dataset.tensors
    if show_plots:
        plots.plot_all_mfs(model, x)
    print('### Testing for {} cases'.format(x.shape[0]))
    y_pred = model(x)
    mse, rmse, perc_loss = calc_error(y_pred, y_actual)
    with torch.no_grad():
        for x, y_actual in data:
            pred = model(x)
            test_loss += F.mse_loss(pred, y_actual).item()
    print('MS error={:.5f}, RMS error={:.5f}, percentage={:.2f}%'
          .format(mse, rmse, perc_loss))
    print(f"Avg loss: {test_loss:>8f} \n")
    if show_plots:
        plots.plotResults(y_actual, y_pred)

#обучение НС
```

```
def train_anfis_with(model, linModel, data, optimizer, criterion, epochs=500,
show_plots=False, error_plot=False, plotNN=False, errorNN=False):
    errors = []  # для графика
    correct = 0
    print('### Training for {} epochs, training size = {} cases'.
          format(epochs, data.dataset.tensors[0].shape[0]))
    for t in range(epochs):
        for x, y_actual in data:
            optimizer.zero_grad()
            y_pred = model.forward(x)
            loss = criterion(y_pred, y_actual)
            loss.backward()
            torch.nn.utils.clip_grad_norm(model.parameters(),max_norm=0.7)
            optimizer.step()
        #Конец эпохи:
        x, y_actual = data.dataset.tensors
        with torch.no_grad():
            model.fit_coeff(x, y_actual)
            y_pred = model.forward(x)
            mse, rmse, perc_loss = calc_error(y_pred, y_actual)
            train_loss = loss.item()
            correct += (y_pred.argmax(1) == y_actual).type(torch.float).sum().item()
        errors.append(perc_loss)
        # Print some progress information as the net is trained:
        if epochs < 30 or t % 10 == 0:
            print('epoch {:4d}: MSE={:.5f}, RMSE={:.5f} ={:.2f}%'
                  .format(t, mse, rmse, perc_loss))
            print(f"loss: {train_loss:>7f}  correct: {correct:>7f}")
        if perc_loss < 2:
            break
    #Конец обучения, вывод графиков при необходимости:
    if show_plots:
        plots.plotResults(y_actual, y_pred, "Результаты ННС")
    if error_plot:
        plots.plotErrors(errors, "Ошибка ННС")
    plt.show()
    with torch.autograd.set_detect_anomaly(True):
        y_pred, error = linearNet.train(linModel, criterion, y_pred, y_actual, plotNN,
errorNN,epochs)
    return y_pred, error

def train_anfis(model, data, n_cons_layer, epochs=500, show_plots=False):
    optimizer = torch.optim.SGD(model.parameters(), lr=1e-4, momentum=0.99)
    criterion = torch.nn.MSELoss(reduction='sum')
```

```
    train_anfis_with(model, data, optimizer, criterion, n_cons_layer, epochs,
show_plots)
```

Функции принадлежности.

```
import torch
from Model.ANFIS_2.anfis_net.anfis import AnfisNet

def _mk_param(val):
    '''Перевод скалярного значения в параметр torch'''
    if isinstance(val, torch.Tensor):
        val = val.item()
    return torch.nn.Parameter(torch.tensor(val, dtype=torch.float))

class SigmoidMembFunc(torch.nn.Module):
    '''Сигмоидальная функция принадлежности с двумя параметрами a, b (a<b)'''
    def __init__(self, a, b):
        super(SigmoidMembFunc, self).__init__()
        # assert a > b, \
        #     'Sigmoidal parameters: must have a < b.'
        self.register_parameter('a', _mk_param(a))
        self.register_parameter('b', _mk_param(b))
    def forward(self, x):
        val = 1/(1 + torch.exp(-self.a * (x-self.b)))
        return val
    def pretty(self):
        return 'SigmoidMembFunc {} {}'.format(self.a, self.b)

def make_sigmoid_mfs(a, blist):
    '''Возвращает список сигмоидальных функций принадлежности с заданным a
и списком b'''
    return [SigmoidMembFunc(a, b) for b in blist]

class GaussMembFunc(torch.nn.Module):
    '''
        Функция принадлежности Гаусса, определяется двумя параметрами:
        mu - среднее и sigma - стандартное отклонение
    '''
    def __init__(self, mu, sigma):
        super(GaussMembFunc, self).__init__()
        self.register_parameter('mu', _mk_param(mu))
        self.register_parameter('sigma', _mk_param(sigma))

    def forward(self, x):
```

```
        val = torch.exp(-torch.pow(x - self.mu, 2) / (2 * self.sigma**2))
        return val

    def pretty(self):
        return 'GaussMembFunc {} {}'.format(self.mu, self.sigma)

def make_gauss_mfs(sigma, mu_list):
    '''Возвращает список функций принадлежности Гаусса
     с заданным стандартным отклонением и списком средних'''
    return [GaussMembFunc(mu, sigma) for mu in mu_list]

class TrapezoidalMembFunc(torch.nn.Module):
    '''
        Трапециевидная функция принадлежности, определяется четырьмя параметрами:
            левее от a: всегда 0
            от a до b: возрастает от 0 до 1
            от b до c: всегда 1
            от c до d: убывает от 1 до 0
            правее d: всегда 0
    '''
    def __init__(self, a, b, c, d):
        super(TrapezoidalMembFunc, self).__init__()
        assert a <= b and b <= c and c <= d,\
            'Trapezoidal parameters: must have a <= b <= c <= d.'
        self.register_parameter('a', _mk_param(a))
        self.register_parameter('b', _mk_param(b))
        self.register_parameter('c', _mk_param(c))
        self.register_parameter('d', _mk_param(d))

    def forward(self, x):
        return torch.where(
            self.a < x,
            torch.where(
                x <= self.b,
                (x - self.a) / (self.b - self.a),
                torch.where(
                    x <= self.c,
                    torch.ones_like(x, requires_grad=True),
                    torch.where(
                        x < self.d,
                        (self.d - x) / (self.d - self.c),
                        torch.zeros_like(x, requires_grad=True)
                    )
```

```
            )
        ),
        torch.zeros_like(x, requires_grad=True)
    )

def make_trap_mfs(a, mlist, width, d):
    '''Возвращает список трапециевидных функций принадлежности, с (a, d, width), и списком центров'''
    return [TrapezoidalMembFunc(a, m-(width/2), m+(width/2), d) for m in mlist]

# Make the classes available via (controlled) reflection:
get_class_for = {n: globals()[n]
                 for n in ['SigmoidMembFunc',
                           'GaussMembFunc',
                           'TrapezoidalMembFunc',
                           ]}

def make_anfis(x, mfs=['Сигмоидальная функция', 'Функция Гаусса',
'Трапециевидная функция'], num_mfs=5, num_out=1, hybrid=True):
    minvals, _ = torch.min(x, dim=0)
    maxvals, _ = torch.max(x, dim=0)
    ranges = maxvals - minvals
    invars = []
    for i in range(len(mfs)):
        if mfs[i] == 'Функция Гаусса':
            sigma = ranges[i] / num_mfs
            mulist = torch.linspace(minvals[i], maxvals[i], num_mfs).tolist()
            invars.append(('x{}'.format(i), make_gauss_mfs(sigma, mulist)))
        if mfs[i] == 'Сигмоидальная функция':
            a = ranges[i] / num_mfs
            blist = torch.linspace(minvals[i], maxvals[i], num_mfs).tolist()
            for b in blist:
                if b < a.item():
                    b = a + (a-b)
            invars.append(('x{}'.format(i), make_sigmoid_mfs(a, blist)))
        if mfs[i] == 'Трапециевидная функция':
            a = minvals[i]
            d = maxvals[i]
            width = ranges[i] / (num_mfs + 2)
            mlist = torch.linspace(minvals[i], maxvals[i], num_mfs + 2).tolist()
            mlist.pop(0)
            mlist.pop(len(mlist) - 1)
            invars.append(('x{}'.format(i), make_trap_mfs(a, mlist, width, d)))
    outvars = ['y{}'.format(i) for i in range(num_out)]
```

```
    model = AnfisNet('Simple anfis_net model', invars, outvars, hybrid=hybrid)
    return model
```

Построение графиков.

```
import matplotlib.pyplot as plt
#функции для построение различных графиков

def plotErrors(errors, str, test=False):
    '''
        График ошибок
    '''
    plt.plot(range(len(errors)), errors, '-ro', label='errors')
    plt.ylabel('Ошибка, %')
    if test:
        plt.xlabel('Измерение, номер')
    else:
        plt.xlabel('Эпоха, номер')
    plt.suptitle(str)
    print("Ошибка",errors[len(errors)-1].item())
    return plt.figure()

def plotResults(y_actual, y_predicted, str):
    '''
        График сравнения прогнозов с актуальными данными
    '''
    plt.plot(range(len(y_predicted)), y_predicted.detach().numpy(),
            '-', c = 'r', label='trained')
    plt.plot(range(len(y_actual)), y_actual.numpy(), 'o', c = 'b', label='original')
    plt.ylabel('Коэффициент неравномерности, доли')
    plt.xlabel('Измерение, номер')
    plt.legend(loc='upper left')
    plt.suptitle(str)
    return plt.figure()

def _plot_mfs(var_name, fv, x, str):
    '''
        График функции принадлежности
    '''
    # Sort x so we only plot each x-value once:
    if var_name == "x0":
        var_name = "SCADA"
    if var_name == "x1":
        var_name = "оперативным отчётам"
```

```
    if var_name == "x2":
        var_name = "имитационной модели"
    xsort, _ = x.sort()
    for mfname, yvals in fv.fuzzify(xsort):
        plt.plot(xsort.tolist(), yvals.tolist(), label=mfname)
    plt.xlabel('Взвешенная сумма по {} '.format(var_name, fv.num_mfs))
    plt.ylabel('Значение функции принадлежности')
    plt.legend(bbox_to_anchor=(1., 0.95))
    plt.suptitle(str)
    return plt.figure()

def plot_all_mfs(model, x, str):
    for i, (var_name, fv) in enumerate(model.layer.fuzzify.varmfs.items()):
        _plot_mfs(var_name, fv, x[:, i], str)
```

Прогнозирование.

```
import torch
import numpy as np
from torch.utils.data import TensorDataset, DataLoader
from Model.ANFIS_2.anfis_net.membership import make_anfis
from Model.ANFIS_2.anfis_net import experimental
from Model.ANFIS_2.anfis_net import plots
from Model.ANFIS_2.anfis_net import linearNet
from torch.nn.modules.module import _addindent
import matplotlib.pyplot as plt

def get_data(name, in_feat=3, batch_size=1024): #чтение данных из файла
    data = np.loadtxt(name)
    x = torch.Tensor(data[:,0:in_feat])
    y = torch.Tensor(data[:,in_feat]).unsqueeze(1)
    td = TensorDataset(x, y)
    print(data)
    print(x)
    print(y)
    return DataLoader(td, batch_size=batch_size, shuffle=False)

def num_cat_correct(model, x, y_actual): #для проверки на корректность
    y_pred = model(x)
    # Change the y-value scores back into 'best category':
    cat_act = torch.argmax(y_actual, dim=1)
    cat_pred = torch.argmax(y_pred, dim=1)
    num_correct = torch.sum(cat_act == cat_pred)
    return num_correct.item(), len(x)
```

```
def train_hybrid(name, in_feat=3, mfs=['Функция Гаусса','Сигмоидальная
функция','Сигмоидальная функция'], num_mfs=3, neurons=10, layers=3,
                 actF=['Sigmoid','Sigmoid'], epochs=400,
                 show_plot_NFS=True, show_plot_mfs=True, error_plot=True,
plotNN=True, errorNN=True): #гибридное обучение
    train_data = get_data(name, in_feat)
    x, y_actual = train_data.dataset.tensors
    model = make_anfis(x, mfs=mfs, num_mfs=num_mfs, num_out=1, hybrid=True)
    linNet = linearNet.LinearNet(neurons, n_hidden_layers=layers, actF=actF)
    #optimizer = torch.optim.SGD(model.parameters(), lr=0.001, momentum=0.99)
#оптимизатор из PyTorch
    optimizer = torch.optim.Rprop(model.parameters(), lr=1e-4)
    #criterion = torch.nn.MSELoss(reduction='sum')
    def criterion(pred, target):
        squares = (pred - target) ** 2
        return squares.mean()
    y_pred, error = experimental.train_anfis_with(model, linNet, train_data, optimizer,
criterion, epochs=epochs, show_plots=show_plot_NFS, error_plot=error_plot,
plotNN=plotNN, errorNN=errorNN)
    if show_plot_mfs:
        plots.plot_all_mfs(model, x, "Функции принадлежности") #графики всех
функций принадлежности для каждой переменной
    trainedNFS = linearNet.TrainedNFS(model, linNet)
    plt.show()
    return trainedNFS, error
def get_data_test(name, in_feat=3): #прогнозирование
    data = np.loadtxt(name)
    x = torch.Tensor(data[:, 0:data.shape[1]-1])
    return x
```

Приложение В

Структура обученной нейро-нечёткой сети:

TrainedNFS(

(anfis): AnfisNet(

Rule 0: IF x0 is mf0 and x1 is mf0 and x2 is mf0

THEN [[0.11275975406169891, 0.12134233862161636, 0.12134233862161636, 0.1607370227575302]]

Rule 1: IF x0 is mf0 and x1 is mf0 and x2 is mf1

THEN [[0.027033589780330658, 0.028924407437443733, 0.028924407437443733, 0.04095987230539322]]

Rule 2: IF x0 is mf0 and x1 is mf0 and x2 is mf2

THEN [[6.923494260716078e-29, 1.372208303507662e-15, 1.372208303507662e-15, 3.314366244143467e-29]]

Rule 3: IF x0 is mf0 and x1 is mf1 and x2 is mf0

THEN [[0.027033589780330658, 0.028924407437443733, 0.028924407437443733, 0.04095987230539322]]

Rule 4: IF x0 is mf0 and x1 is mf1 and x2 is mf1

THEN [[0.006758792791515589, 0.007092977873980999, 0.007092977873980999, 0.012749732472002506]]

Rule 5: IF x0 is mf0 and x1 is mf1 and x2 is mf2

THEN [[1.7444724361457162e-29, 1.372208303507662e-15, 1.372208303507662e-15, 8.315860059411967e-30]]

Rule 6: IF x0 is mf0 and x1 is mf2 and x2 is mf0

THEN [[6.923494260716078e-29, 1.372208303507662e-15, 1.372208303507662e-15, 3.314366845996575e-29]]

Rule 7: IF x0 is mf0 and x1 is mf2 and x2 is mf1

THEN [[1.7444724361457162e-29, 1.372208303507662e-15, 1.372208303507662e-15, 8.315860811728351e-30]]

Rule 8: IF x0 is mf0 and x1 is mf2 and x2 is mf2

THEN [[8.804091353523114e-13, 8.804091353523114e-13, 8.804091353523114e-13, 8.804091353523114e-13]]

Rule 9: IF x0 is mf1 and x1 is mf0 and x2 is mf0

THEN [[0.4503750205039978, 0.47834718227386475, 0.47834718227386475, 0.40002456307411194]]

Rule 10: IF x0 is mf1 and x1 is mf0 and x2 is mf1

THEN [[0.11016902327537537, 0.11672276258468628, 0.11672276258468628, 0.09901382774114609]]

Rule 11: IF x0 is mf1 and x1 is mf0 and x2 is mf2

THEN [[8.921792659804117e-27, 1.372208303507662e-15, 1.372208303507662e-15, 3.697584426137299e-27]]

Rule 12: IF x0 is mf1 and x1 is mf1 and x2 is mf0

THEN [[0.11016902327537537, 0.11672275513410568, 0.11672275513410568, 0.09901382774114609]]

Rule 13: IF x0 is mf1 and x1 is mf1 and x2 is mf1

THEN [[0.027273284271359444, 0.028726622462272644, 0.028726622462272644, 0.02695039100944996]]

Rule 14: IF x0 is mf1 and x1 is mf1 and x2 is mf2

THEN [[2.2700262179011813e-27, 1.372208303507662e-15, 1.372208303507662e-15, 9.40222532148824e-28]]

Rule 15: IF x0 is mf1 and x1 is mf2 and x2 is mf0

THEN [[8.921792659804117e-27, 1.372208303507662e-15, 1.372208303507662e-15, 3.697584426137299e-27]]

Rule 16: IF x0 is mf1 and x1 is mf2 and x2 is mf1

THEN [[2.270026025308187e-27, 1.372208303507662e-15, 1.372208303507662e-15, 9.402223395558296e-28]]

Rule 17: IF x0 is mf1 and x1 is mf2 and x2 is mf2

THEN [[8.804091353523114e-13, 8.804091353523114e-13, 8.804091353523114e-13, 8.804091353523114e-13]]

Rule 18: IF x0 is mf2 and x1 is mf0 and x2 is mf0

THEN [[8.804091353523114e-13, 8.804091353523114e-13, 8.804091353523114e-13, 8.804091353523114e-13, 8.804091353523114e-13]]

Rule 19: IF x0 is mf2 and x1 is mf0 and x2 is mf1

THEN [[8.804091353523114e-13, 8.804091353523114e-13, 8.804091353523114e-13, 8.804091353523114e-13, 8.804091353523114e-13]]

Rule 20: IF x0 is mf2 and x1 is mf0 and x2 is mf2

THEN [[8.804091353523114e-13, 8.804091353523114e-13, 8.804091353523114e-13, 8.804091353523114e-13, 8.804091353523114e-13]]

Rule 21: IF x0 is mf2 and x1 is mf1 and x2 is mf0

THEN [[8.804091353523114e-13, 8.804091353523114e-13, 8.804091353523114e-13, 8.804091353523114e-13, 8.804091353523114e-13]]

Rule 22: IF x0 is mf2 and x1 is mf1 and x2 is mf1

THEN [[8.804091353523114e-13, 8.804091353523114e-13, 8.804091353523114e-13, 8.804091353523114e-13, 8.804091353523114e-13]]

Rule 23: IF x0 is mf2 and x1 is mf1 and x2 is mf2

THEN [[8.804091353523114e-13, 8.804091353523114e-13, 8.804091353523114e-13, 8.804091353523114e-13, 8.804091353523114e-13]]

Rule 24: IF x0 is mf2 and x1 is mf2 and x2 is mf0

THEN [[8.804091353523114e-13, 8.804091353523114e-13, 8.804091353523114e-13, 8.804091353523114e-13, 8.804091353523114e-13]]

Rule 25: IF x0 is mf2 and x1 is mf2 and x2 is mf1

THEN [[8.804091353523114e-13, 8.804091353523114e-13, 8.804091353523114e-13, 8.804091353523114e-13, 8.804091353523114e-13]]

Rule 26: IF x0 is mf2 and x1 is mf2 and x2 is mf2

THEN [[8.804091353523114e-13, 8.804091353523114e-13, 8.804091353523114e-13, 8.804091353523114e-13, 8.804091353523114e-13]]

(layer): ModuleDict(

(fuzzify): Input variables

Variable x0

```
      - mf0: GaussMembFunc(mu=-0.260710746049881,
sigma=0.7011358737945557)
      - mf1: GaussMembFunc(mu=-0.5447078943252563,
sigma=1.4261817932128906)
      - mf2: GaussMembFunc(mu=3.6753194332122803, sigma=-
0.06469782441854477)
      Variable x1
      - mf0: SigmoidMembFunc(a=13.796218872070312, b=0.20123261213302612)
      - mf1: SigmoidMembFunc(a=0.0760587751865387, b=16.591962814331055)
      - mf2: SigmoidMembFunc(a=5.121467113494873, b=13.821925163269043)
      Variable x2
      - mf0: SigmoidMembFunc(a=13.796218872070312, b=0.20123261213302612)
      - mf1: SigmoidMembFunc(a=0.0760587751865387, b=16.591962814331055)
      - mf2: SigmoidMembFunc(a=5.121467113494873, b=13.821925163269043)
      (rules): AntecedentLayer()
      (consequent): ConsequentLayer()
      (weighted_sum): WeightedSumLayer()
    )
  )
  (linearNet): LinearNet(
    (fc1): Linear(in_features=1, out_features=10, bias=True)
    (fc2): Linear(in_features=10, out_features=10, bias=True)
    (fc3): Linear(in_features=10, out_features=1, bias=True)
    (actS): Sigmoid()
    (actG): GELU()
    (actR): ReLU()
  )
)
```

Схема параметров для линейной части НИС:

Layer (type)	Output Shape	Param #
Linear-1	[-1, 43, 10]	20
Sigmoid-2	[-1, 43, 10]	0
Linear-3	[-1, 43, 10]	110
Sigmoid-4	[-1, 43, 10]	0
Linear-5	[-1, 43, 1]	11

Приложение Г

Программный код алгоритма формирования оптимального графика ППР:

```
 def unit_list(self, T_end, T_start=datetime.date.today()):
  unit_inds = []
  for i in range(len(self._units)):
   if self._last_time_calc < self._units[i].forecast_date < T_end:
    unit_inds.append(i)
   elif self._units[i].forecast_date < T_end:
    if abs(self._units[i].forecast_date - datetime.date.today()) < self._units[i].deviation:
     self._warning_list.append(self._units[i])
    else:
     unit_inds.append(i)
  return unit_inds
 def unit_distribution(self, unit_inds):
  T_query = datetime.date.today()
  k = len(self._repairs)
  n = len(self._units)
  for i in range(k):
   if self._repairs[i].date >= T_query:
    j = i
    break
  self.j = j
  self.arr = np.zeros((n, k-j))
  for i in unit_inds:
   Min0 = 1000000
   MinInd = 0
   for z in range(j, k):
    Min = abs(self._units[i].forecast_date - self._repairs[z].date).days
    if Min <= Min0:
```

```
        Min0 = Min
        MinInd = z
    if self._units[i].forecast_date >= self._repairs[MinInd].date:
      self._units[i].repair_index = MinInd
      self.arr[i, MinInd-j] = 1
    elif Min0 <= self._units[i].deviation:
      self._units[i].repair_index = MinInd
      self.arr[i, MinInd-j] = 1
    else:
      self._units[i].repair_index = MinInd-1
      self.arr[i, MinInd-1-j] = 1
def calculate_K(self, unit_inds, arr=[]):
  if len(arr)==0:
    arr  = np.copy(self.arr)
  K2 = 0
  for i in unit_inds:
    K2 += self._units[i].price
  self._time = np.sum(arr)
  S_per_hour = self._V*self._S/self._T
  K1 = self._time*S_per_hour
  return K1, K2
def optimal_plan(self, unit_inds):
  for i in unit_inds:
    j = self._units[i].repair_index
    repair = self._repairs[j]
    if self._units[i].assambly_status is True:
      self.arr[i][j] = self._units[i].repair_time_group
    else:
      self.arr[i][j] = self._units[i].repair_time_alone
  self._K1, self._K2 = self.calculate_K(unit_inds)
```

```
def optimal_distribution(self, unit_inds):
  iter=0
  for i in unit_inds:
    k = self._units[i].repair_index
    if k!=0:
      if self.arr[i][k] == self._units[i].repair_time_alone:
        for j in np.nonzero(self.arr[:, k-1])[0]:
          if self._units[i].element_id == self._units[j].element_id:
            iter+=1
            arr_temp = np.copy(self.arr)
            arr_temp[j][k-1] = self._units[j].repair_time_group
            arr_temp[i][k-1] = self._units[i].repair_time_group
            arr_temp[i][k] = 0
            self._units[i].repair_index = k-1
            print(str(iter), 'итерация по оптимизации')
            self.show_plan(arr_temp)
            K1opt, K2opt = self.calculate_K(unit_inds, arr_temp)
            print('K1 =', K1opt)
            print('K2 =', K2opt)
            print('K =', K1opt+K2opt)
            if (K1opt + K2opt) < (self._K1 + self._K2):
              self._K1 = K1opt
              self._K2 = K2opt
              self.arr = np.copy(arr_temp)
            else:
              self._units[i].repair_index = k
            break
def CREATE_OPTIMAL_PLAN(self, T_end):
  unit_inds = self.unit_list(T_end)
      if len(self._warning_list) != 0:
```

```
    print('Подлежат срочному ремонту:')
    for unit in self._warning_list:
      print(unit.name, 'Прогнозная дата выработки ресурса: ', unit.forecast_date)
  self.unit_distribution(unit_inds)
  print('Распределение по ближайшим датам')
  self.show_plan()
  print('Псевдооптимальный план')
  self.optimal_plan(unit_inds)
  self.show_plan()
  print('K1 =', self._K1)
  print('K2 =', self._K2)
  print('K =', self._K1+self._K2)
  self.optimal_distribution(unit_inds)
  print('Окончательный план')
  self.show_plan()
  print('K1 =', self._K1)
  print('K2 =', self._K2)
  print('K =', self._K1+self._K2)
  self._last_time_calc = T_end
def show_plan(self, arr=[]):
  if len(arr)==0:
    arr=self.arr
  indices = [unit.name for unit in self._units]
  headers = [repair.date.strftime('%d.%m.%Y') for repair in self._repairs]
  df = pd.DataFrame(arr, index=indices, columns=headers)
  display(df.loc[(df!=0).any(axis=1)])
```

Приложение Д

Таблица Д.1 – Схема таблицы Platform

Атрибут	**Тип**	**Размер**	**Допустимые значения**	**Значение по умолчанию**	**Ключи**
id	целочисл.	4 Б	уникал., не пустое	-	PK: id;
name	симв.	30 Б	уникал. , не пустое	-	
description	симв.	100 Б	-	NULL	
file_location	симв.	200 Б	-	NULL	

Таблица Д.2 – Схема таблицы Department

Атрибут	**Тип**	**Размер**	**Допустимые значения**	**Значение по умолчанию**	**Ключи**
id	целочисл.	4 Б	уникал. , не пустое		PK: id, FK: platform_id ref. Platform(id)
name	симв.	30 Б	уникал. , не пустое		
description	симв.	100 Б	-	NULL	
platform_id	целочисл.	4 Б	не пустое		
S	веществ.	4 Б	> 0, не пустое		
V	веществ.	4 Б	> 0, не пустое		
T	веществ.	4 Б	> 0, не пустое		
file_location	симв.	200 Б	-	NULL	

Таблица Д.3 – Схема таблицы Section

Атрибут	**Тип**	**Размер**	**Допустимые значения**	**Значение по умолчанию**	**Ключи**
id	целочисл.	4 Б	уникал., не пустое		PK: id, FK: department_id ref. Department(id)
name	симв.	30 Б	уникал., не пустое		
description	симв.	100 Б	-	NULL	
department_id	целочисл.	4 Б	не пустое		
file_location	симв.	200 Б	-	NULL	

Таблица Д.4 – Схема таблицы Object

Атрибут	**Тип**	**Размер**	**Допустимые значения**	**Значение по умолчанию**	**Ключи**
id	целочисл.	4 Б	уникал., не пустое		PK: id, FK: section_id ref. Section(id)
code	симв.	10 Б	уникал., не пустое		
name	симв.	30 Б	не пустое		
description	симв.	100 Б	-	NULL	
section_id	целочисл.	4 Б	не пустое		
file_location	симв.	200 Б	-	NULL	
file_location	симв.	200 Б	-	NULL	

Таблица Д.5 – Схема таблицы Class

Атрибут	**Тип**	**Размер**	**Допустимые значения**	**Значение по умолчанию**	**Ключи**
id	целочисл.	4 Б	уникал., не пустое		
name	симв.	30 Б	уникал., не пустое		
description	симв.	100 Б	-	NULL	
model_path	симв.	200 Б	-	NULL	

Таблица Д.6 – Схема таблицы Element

Атрибут	**Тип**	**Размер**	**Допустимые значения**	**Значение по умолчанию**	**Ключи**
id	целочисл.	4 Б	уникал., не пустое		PK: id; FK: object_id ref. Object(id)
name	симв.	30 Б	не пустое		
description	симв.	100 Б	-	NULL	
object_id	целочисл.	4 Б	не пустое		
file_location	симв.	200 Б	-	NULL	

Таблица Д.7 – Схема таблицы Detail

Атрибут	**Тип**	**Размер**	**Допустимые значения**	**Значение по умолчанию**	**Ключи**
id	целочисл.	4 Б	уникал., не пустое		PK: id; FK: class_id ref. Class(id), element_id ref. Element(id)
name	симв.	30 Б	не пустое		
description	симв.	100 Б	-	NULL	
class_id	целочисл.	4 Б	не пустое		
element_id	целочисл.	4 Б	не пустое		
file_location	симв.	200 Б	-	NULL	

Таблица Д.8 – Схема таблицы BIDZIP

Атрибут	**Тип**	**Размер**	**Допустимые значения**	**Значение по умолчанию**	**Ключи**
id	целочисл.	4 Б	уникал., не пустое		PK: id; FK: detail_id ref. Detail(id)
name	симв.	30 Б	не пустое		
description	симв.	100 Б	-	NULL	
detail_id	целочисл.	4 Б	не пустое		
date_install	дата	4 Б	< текущ. дата, не пустое		
price	веществ.	4 Б	> 0, не пустое		
currenttime	веществ.	4 Б	-	0	
criticaltime	веществ.	4 Б	> 0, не пустое		

Таблица Д.9 – Схема таблицы Sensor

Атрибут	**Тип**	**Размер**	**Допустимые значения**	**Значение по умолчанию**	**Ключи**
id	целочисл.	4 Б	уникал., не пустое		PK: id;
name	симв.	30 Б	уникал., не пустое		
description	симв.	100 Б	-	NULL	
SCADA	булевый	1 Б	не пустое	TRUE	
hand	булевый	1 Б	не пустое	FALSE	
imitation	булевый	1 Б	не пустое	FALSE	

Таблица Д.10 – Схема таблицы Sensor_Detail

Атрибут	**Тип**	**Размер**	**Допустимые значения**	**Значение по умолчанию**	**Ключи**
sensor_id	целочисл.	4 Б	не пустое		FK: detail_id ref. Detail(id), sensor_id ref. Sensor(id)
detail_id	целочисл.	4 Б	не пустое		
nominal	веществ.	4 Б	> 0, не пустое	1	
nns_input	целое	4 Б	> 0	NULL	

Таблица Д.11 – Схема таблицы SCADA

Атрибут	**Тип**	**Размер**	**Допустимые значения**	**Значение по умолчанию**	**Ключи**
time	дата и время	8 Б	< текущ. дата и время, не пустое		FK: sensor_id ref Sensor(id)
val	веществ.	4 Б	не пустое	0	
sensor_id	целочисл.	4 Б	не пустое		

Таблица Д.12 – Схема таблицы Hand

Атрибут	**Тип**	**Размер**	**Допустимые значения**	**Значение по умолчанию**	**Ключи**
time	дата и время	8 Б	< текущ. дата и время, не пустое		FK: sensor_id ref Sensor(id)
val	веществ.	4 Б	не пустое	0	
sensor_id	целочисл.	4 Б	не пустое		

Таблица Д.13 – Схема таблицы Imitation

Атрибут	**Тип**	**Размер**	**Допустимые значения**	**Значение по умолчанию**	**Ключи**
time	дата и время	8 Б	< текущ. дата и время, не пустое		FK: sensor_id ref Sensor(id)
val	веществ.	4 Б	не пустое	0	
sensor_id	целочисл.	4 Б	не пустое		

Таблица Д.14 – Схема таблицы NNS_output

Атрибут	**Тип**	**Размер**	**Допустимые значения**	**Значение по умолчанию**	**Ключи**
time	дата и время	8 Б	< текущ. дата и время, не пустое		FK: detail_id ref Detail(id)
val	веществ.	4 Б	не пустое		
detail_id	целочисл.	4 Б	не пустое		

Таблица Д.15 – Схема таблицы Journal_table

Атрибут	**Тип**	**Размер**	**Допустимые значения**	**Значение по умолчанию**	**Ключи**
id	целочисл.	4 Б	уникал., не пустое		PK: id
title	симв.	30 Б	не пустое		
event_time	дата и время	8 Б	< текущ. дата и время, не пустое		

Таблица Д.16 – Схема таблицы Journal_table_Detail

Атрибут	**Тип**	**Размер**	**Допустимые значения**	**Значение по умолчанию**	**Ключи**
journal_id	целочисл.	4 Б	не пустое		FK: journal_id ref. Journal_table(id), detail_id ref. Detail(id)
detail_id	целочисл.	4 Б	не пустое		
description	симв.	200 Б	-	NULL	

Таблица Д.17 – Схема таблицы Repair_type

Атрибут	**Тип**	**Размер**	**Допустимые значения**	**Значение по умолчанию**	**Ключи**
id	целочисл.	4 Б	уникал., не пустое		PK: id
name	симв.	30 Б	уникал., не пустое		

Таблица Д.18 – Схема таблицы Repair_type_BIDZIP

Атрибут	**Тип**	**Размер**	**Допустимые значения**	**Значение по умолчанию**	**Ключи**
id	целочисл.	4 Б	уникал., не пустое		PK: id; FK: repair_type_id ref. Repair_type(id), bidzip_id ref. BIDZIP(id)
repair_type_id	целочисл.	4 Б	не пустое		
bidzip_id	целочисл.	4 Б	не пустое		
time_group	веществ.	4 Б	> 0, не пустое		
time_alone	веществ.	4 Б	> 0, не пустое		

Таблица Д.19 – Схема таблицы Repair_Complex

Атрибут	**Тип**	**Размер**	**Допустимые значения**	**Значение по умолчанию**	**Ключи**
id	целочисл.	4 Б	уникал., не пустое		PK: id; FK: bidzip_id ref. BIDZIP(id)
name	симв.	30 Б	уникал., не пустое		
bidzip_id	целочисл.	4 Б	не пустое		

Таблица Д.20 – Схема таблицы Repair_complex_to_Type

Атрибут	**Тип**	**Размер**	**Допустимые значения**	**Значение по умолчанию**	**Ключи**
id_repair_type_bidzip	целочисл.	4 Б	не пустое		FK: id_repair_type_bidzip ref. Repair_type_BIDZIP(id), id_retair_complex ref. Repair_complex(id)
id_repair_complex	целочисл.	4 Б	не пустое		

Таблица Д.21 – Схема таблицы Schedule

Атрибут	**Тип**	**Размер**	**Допустимые значения**	**Значение по умолчанию**	**Ключи**
id	целочисл.	4 Б	уникал., не пустое		PK: id
forming_date	дата и время	8 Б	уникал.		

Таблица Д.22 – Схема таблицы Schedule_Repair_Complex

Атрибут	**Тип**	**Размер**	**Допустимые значения**	**Значение по умолчанию**	**Ключи**
schedule_id	целочисл.	4 Б	не пустое		FK:schedule_id ref. Schedule(id), repair_complex_id ref. Repair_Complex(id)
repair_complex_id	целочисл.	4 Б	не пустое		
date_remove	дата	4 Б	не пустое		
time_plan	веществ.	4 Б	> 0	NULL	
time_fact	веществ.	4 Б	> 0	NULL	

Таблица Д.23 – Схема таблицы Post

Атрибут	**Тип**	**Размер**	**Допустимые значения**	**Значение по умолчанию**	**Ключи**
id	целочисл.	4 Б	уникал., не пустое		PK: id
name	симв.	30 Б	уникал., не пустое		

Таблица Д.24 – Схема таблицы User_table

Атрибут	**Тип**	**Размер**	**Допустимые значения**	**Значение по умолчанию**	**Ключи**
id	целочисл.	4 Б	уникал., не пустое		PK: id; FK: post_id ref. Post(id)
login	симв.	30 Б	уникал., не пустое		
password	симв.	20 Б	> 8 симв.		
name	симв.	20 Б	не пустое		
surname	симв.	30 Б	не пустое		
fathername	симв.	30 Б	-	NULL	
post_id	целочисл.	4 Б	не пустое		
platform_id	целочисл.	4 Б	не пустое		

Таблица Д.25 – Схема таблицы Role

Атрибут	**Тип**	**Размер**	**Допустимые значения**	**Значение по умолчанию**	**Ключи**
id	целочисл.	4 Б	уникал.		PK: id
name	симв.	30 Б	уникал.		

Таблица Д.26 – Схема таблицы User_Role

Атрибут	**Тип**	**Размер**	**Допустимые значения**	**Значение по умолчанию**	**Ключи**
user_id	целочисл.	4 Б	не пустое		FK: user_id ref. User_table(id), role_id ref. Role(id)
role_id	целочисл.	4 Б	не пустое		

Таблица Д.27 – Схема таблицы VisitLog

Атрибут	**Тип**	**Размер**	**Допустимые значения**	**Значение по умолчанию**	**Ключи**
user_id	целочисл.	4 Б	уникал.		FK: user_id ref. User_table(id)
login_time	дата и время	8 Б	не пустое		
logout_time	дата и время	8 Б	-	NULL	
pc_name	симв.	30 Б	-	NULL	